目 录



第1章　计算机数学语言概述

1.1　数学问题计算机求解概述

1.1.1　为什么要学习计算机数学语言

求解数学问题时手工推导当然是有用的，但并不是所有的问题都是能手工推导的，故需要由计算机来完成相应的任务。用计算机的方式也有两种，其一是用成型的数值分析算法、数值软件包与手工编程的方法相结合的求解方法，其二是采用国际上有影响的专门计算机语言来求解问题，这类语言包括 MATLAB、Mathematica、Maple 等，本书统一称之为**计算机数学语言**。顾名思义，用数值方法只能求解数值计算的问题，至于像公式推导等数学问题，例如求解 $x^3+ax+c=d$ 方程的解，在 a,c,d 不是给定数值时，数值分析的方式是没有用的，必须使用计算机数学语言来求解。

在系统介绍本书的内容之前，先介绍几个例子，读者可以思考其中提出的问题，从中体会学习本书的必要性。

【例 1-1】大学的高等数学课程学习了微分与积分的概念和数学推导方法，但实际应用中可能遇到高阶导数的问题。已知 $f(x)=\sin x/(x^2+4x+3)$ 这样的简单函数，如何求解出 $\mathrm{d}^4f(x)/\mathrm{d}x^4$？当然，用手工推导是可行的，由高等数学的知识先得出 $\mathrm{d}f(t)/\mathrm{d}x$，对结果求导数得出二阶导数，对结果再求导得出三阶导数，对其再求导一步就能求出所需的 $\mathrm{d}^4f(x)/\mathrm{d}x^4$，重复此方法还能求出更高阶的导数。这个过程比较机械，适合计算机实现，所以用现有的计算机数学语言可以由一条语句得出结果为

$$\begin{aligned}\frac{\mathrm{d}^4f(t)}{\mathrm{d}t^4}=&\frac{\sin x}{x^2+4x+3}+4\frac{\cos x\,(2x+4)}{(x^2+4x+3)^2}-12\frac{\sin x\,(2x+4)^2}{(x^2+4x+3)^3}+12\frac{\sin x}{(x^2+4x+3)^2}-24\frac{\cos x\,(2x+4)^3}{(x^2+4x+3)^4}\\&+48\frac{\cos x\,(2x+4)}{(x^2+4x+3)^3}+24\frac{\sin x\,(2x+4)^4}{(x^2+4x+3)^5}-72\frac{\sin x\,(2x+4)^2}{(x^2+4x+3)^4}+24\frac{\sin x}{(x^2+4x+3)^3}\end{aligned}$$

当然，经过计算机化简，还可以得出更简的形式为

$$\begin{aligned}\frac{\mathrm{d}^4f(x)}{\mathrm{d}x^4}=&8(x^5+10x^4+26x^3-4x^2-99x-102)\frac{\cos x}{(x^2+4x+3)^4}+\\&(x^8+16x^7+72x^6-32x^5-1094x^4-3120x^3-3120x^2+192x+1581)\frac{\sin x}{(x^2+4x+3)^5}\end{aligned}$$

显然，若依赖手工推导，得出这样的结果需要很繁杂、细致的工作，稍有不慎就可能得出错误的结果，由此，手工推导得出结果的可信度有时是值得怀疑的。如果能采用计算机代替手工推导则会既省力，又增加可信度，故需要计算机数学语言来解决这样的问题。实践表明，利用著名的 MATLAB 语言，在 1.5 秒内[①]就可以精确地求出 $\mathrm{d}^{100}f(x)/\mathrm{d}x^{100}$。

[①]本书中涉及的求解时间均指作者使用的 PIII 1.12MHz/256MB 的笔记本计算机上测出的，且 CPU 的不同运行状态下测出的时间也有差异。

【例 1-2】在许多学科的实际应用中经常需要求出多项式方程的根。著名的 Abel 定理已经有了定论，5 次或以上的多项式方程没有通用的解析解求解方法，但在实际应用中经常需要求解高次代数方程的根，故可以采用数值方法求解，如使用林士谔-Bairstrow 算法，这是数值分析中最常见的方法。考虑下面多项式方程

$$s^6+9s^5+\frac{135}{4}s^4+\frac{135}{2}s^3+\frac{1215}{16}s^2+\frac{729}{16}s+\frac{729}{64}=0$$

用林士谔-Bairstrow 算法得出的结果是

$$s_{1,2}=-1.5056\pm \mathrm{j}0.0032,\ \ s_{3,4}=-1.5000\pm \mathrm{j}0.0065,\ \ s_{5,6}=-1.4944\pm \mathrm{j}0.0032$$

将 s_1 代入原始方程，则能容易计算出方程左侧为 $-8.7041\times 10^{-14}-\mathrm{j}1.8353\times 10^{-15}$。虽然误差不大，但采用计算机数学语言能得出更精确的结果，即所有的根均为 -1.5。

【例 1-3】线性代数课程中介绍了求解矩阵行列式的方法，例如用代数余子式的方法可以将一个 n 阶矩阵的行列式问题化简成 n 个 $n-1$ 阶行列式问题，而 $n-1$ 阶的又可以化简为 $n-2$ 阶的问题，这样用递归的方法可以最终化简成一阶矩阵的行列式求解问题，而该问题是有解析解的，就是该一阶矩阵本身，所以数学家可以得出结论，任意阶矩阵的行列式都可以直接求解出解析解。

事实上，这样的结论忽略了复杂度或可行性问题，这样的算法计算量很庞大，高达 $(n-1)(n+1)!+n$，例如 $n=20$ 时，运算次数为 9.7073×10^{20}，相当于在每秒亿次的巨型机 (中国最快的银河计算机) 上 3000 年的计算量，所以虽然用代数余子式的方法可以求解，但求解是不现实的。其实在某些领域中甚至需要求解成百上千阶的问题，所以用代数余子式的方法是不可行的。

数值分析中提供了求解行列式问题的各种各样算法，但传统的方法对某些矩阵有时会得出错误的结果，特别是接近奇异的矩阵。考虑 Hilbert 矩阵

$$\boldsymbol{H}=\begin{bmatrix}1 & 1/2 & 1/3 & \cdots & 1/n\\ 1/2 & 1/3 & 1/4 & \cdots & 1/(n+1)\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1/n & 1/(n+1) & 1/(n+2) & \cdots & 1/(2n-1)\end{bmatrix} \tag{1-1-1}$$

并假设 $n=20$，则用传统的数值分析很容易得出 $\det(\boldsymbol{H})=0$ 的错误结论。事实上，用计算机数学语言 MATLAB 很容易在 0.4 秒内得出该行列式的精确解为

$$\det(\boldsymbol{H})=\frac{1}{\underbrace{2377454716\cdots 36800000000000000000000000000000000000}_{225\text{ 位，因排版限制省略了中间的数字}}}\simeq 4.206179\times 10^{-224}$$

求解一般高阶矩阵求逆问题需要计算机数学语言，对特殊的矩阵问题更需要这样的语言，以免得出错误的结果。

【例 1-4】考虑著名的非线性方程 —— Van der Pol 方程 $\ddot{y}+\mu(y^2-1)\dot{y}+y=0$，当 μ 很大时，例如 $\mu=1000$，传统的数值分析方法求解之可能有问题，需要用专用的刚性方程求解算法进行求解，而不能利用数值分析类课程中介绍的 Runge-Kutta 算法可以求解。

如果一阶微分方程可以写成 $\mathrm{d}y(t)/\mathrm{d}t=-0.1y(t)+0.2y(t-30)/[1+y^{10}(t-30)]$，这样的方程称为延迟微分方程，一般的数值分析教材和软件包中均不提供这样方程的数值解法，所以只能

采用计算机数学语言，如 MATLAB 中的延迟微分方程求解函数 dde23() 或图形化建模仿真工具 Simulink 来求解这样的问题。在本书后面相应的内容中将介绍此方程的解法。

【例 1-5】考虑最优化问题，假设线性规划问题的数学描述如下：

$$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} 2x_2+x_3+4x_4+2x_5 \leqslant 54 \\ 3x_1+4x_2+5x_3-x_4-x_5 \leqslant 62 \\ x_1,x_2 \geqslant 0,,x_3 \geqslant 3.32, x_4 \geqslant 0.678, x_5 \geqslant 2.57 \end{cases}} (-2x_1-x_2-4x_3-3x_4-x_5)$$

因为是有约束问题，不能用高等数学中的令目标函数导数为 0，得出若干方程再求解方程的方式求解最优化问题，而必须用线性规划中介绍的算法求解之，得出 $x_1 = 19.7850$, $x_2 = 0$, $x_3 = 3.3200$, $x_4 = 11.3850$, $x_5 = 2.5700$。

这样的求解借助数值分析或最优化方法等课程介绍的数值算法可以容易地实现。但如果再添加约束，例如需要得出该最优化问题的整数解，原来的问题就变成了整数规划问题。很少有相关书籍、软件能直接求解这样的问题。而利用计算机数学语言可以求出该整数规划问题的解为 $x_1 = 19, x_2 = 0, x_3 = 4, x_4 = 10, x_5 = 5$。

【例 1-6】许多后续课程将用到的高等应用数学分支，如积分变换、复变函数、偏微分方程、数据插值与拟合、概率论与数理统计、数值分析等，课程考试之后您还记得其中问题的求解方法吗?

很多专门的学科，如电路、电子技术、电力电子技术、电机与拖动、自动控制原理等，在介绍原理与方法时一般采用简单的例子，回避高阶的或复杂的例子。究其原因，是当时缺少高水平计算机数学语言甚至是数值分析技术的支持，所以在相关学科中的很多方法不一定适合于复杂的问题求解。在实际研究中遇到稍复杂一点的问题时，只靠手工推导的方法是得不出精确结果的，所以需要特殊的专业软件或语言来解决问题，而计算机数学语言，如 MATLAB 语言，通常可以较好地解决相关问题。

从上面的例子可以看出，解决数学问题用手工推导的方法虽然有时可行，但对很多复杂问题不现实或不可靠，用传统数值分析课程甚至成型的软件包 (如在国际上享有盛誉的 *Numerical Recipes*[39]) 得出的结果有时也是错误的，故需要学习计算机数学语言，以更好地解决以后学习和研究中遇到的问题。

1.1.2 数学问题的解析解与数值解

现代科学与工程的进展离不开数学。数学家们感兴趣的问题和其他科学家、工程技术人员所关注的问题是不同的。数学家往往对数学问题的解析解，或称闭式解 (closed-form solution) 和解的存在性严格证明感兴趣，而工程技术人员一般对如何求出数学问题的解更关心。换句话说，能用某种方法获得问题的解则是工程技术人员更关心的问题。而获得这样解的最直接方法就是通过数值解法技术。

解析解不存在的情况在数学上并不罕见，甚至可以说，这样的现象是常见的。例如，定积分 $\dfrac{2}{\sqrt{\pi}}\displaystyle\int_0^a \mathrm{e}^{-x^2}\mathrm{d}x$ 在上限为有穷时就没有解析解。数学家可以用新的函数 $\mathrm{erf}(a)$ 去定义这样的解，但解的值到底多大却不是一目了然的。所以，在这样的情况下，要想获得积分的值，就必须采用数值解技术。

再例如，圆周率 π 的值本身就没有解析解，中国古代的数学家、天文学家祖冲之早在公元 480 年就算定了该值在 3.1415926 和 3.1415927 之间。在一般科学与工程应用中，取这样的值就能保证较高的精度，而对于粗略估算来说，使用公元前 250 年 (?) 阿基米德的 3.1418 也未尝不可，而没有必要非去追求不存在的解析解不可。所以在这样的问题上，数值解法的优势就显示出来了。

数学问题的数值解法已经成功地应用于各个领域。例如，在力学领域，常用有限元法求解偏微分方程；在航空、航天与自动控制领域，经常用到数值线性代数与常微分方程的数值解法等解决实际问题；在工程与非工程系统的计算机仿真中，核心问题的求解也需要用到各种差分方程、常微分方程的数值解法；在高科技的数字信号处理领域，离散的快速 Fourier 变换 (FFT) 已经成为其不可或缺的工具。在科学工程研究中能掌握一个或多个实用的计算工具，无疑会为研究者提供解决实际问题的强有力手段。

1.1.3 数学运算问题软件包发展概述

数字计算机的出现给数值计算技术的研究注入了新的活力。在数值计算技术的早期发展中，出现了一些著名的数学软件包，如美国的基于特征值的软件包 EISPACK[10, 42]和线性代数软件包 LINPACK[7]，英国牛津数值算法研究组 (Numerical Algorithm Group，NAG) 开发的 NAG 软件包[34]及享有盛誉的著作 *Numerical Recipes*[39]中给出的程序集等，这些都是在国际上广泛流行的、有着较高声望的软件包。

美国的 EISPACK 和 LINPACK 都是基于矩阵特征值和奇异值解决线性代数问题的专用软件包。限于当时的计算机发展状况，这些软件包大都是由 Fortran 语言编写的源程序组成的。

例如，若想求出 N 阶实矩阵 $\boldsymbol{A}$ 的全部特征值 (用 $\boldsymbol{W}_R$, $\boldsymbol{W}_I$ 数组分别表示其实虚部) 和对应的特征向量矩阵 $\boldsymbol{Z}$，则 EISPACK 软件包给出的子程序建议调用路径为

```
CALL BALANC(NM,N,A,IS1,IS2,FV1)
CALL ELMHES(NM,N,IS1,IS2,A,IV1)
CALL ELTRAN(NM,N,IS1,IS2,A,IV1,Z)
CALL HQR2(NM,N,IS1,IS2,A,WR,WI,Z,IERR)
IF (IERR.EQ.0) GOTO 99999
CALL BALBAK(NM,N,IS1,IS2,FV1,N,Z)
```

由上面的叙述可以看出，要求取矩阵的特征值和特征向量，首先要给一些数组和变量依据 EISPACK 的格式作出定义和赋值，并编写出主程序，再经过编译和连接过程，形成可执行文件，最后才能得出所需的结果。

英国的 NAG 软件包和美国学者的 *Numerical Recipes* 工具包则包括了各种各样数学问题的数值解法，二者中 NAG 的功能尤其强大。NAG 的子程序都是以字母加数字编号的形式命名的，非专业人员很难找到适合自己问题的子程序，更不用说能保证以正确的格式去调用这些子程序了。这些程序包使用起来极其复杂，谁也不能保证不发生错误，NAG 数百页的使用手册就有十几本之多！

Numerical Recipes 一书中给出的一系列算法语言源程序也是一个在国际上广泛应用

的软件包。该书中的子程序有 C, Fortran 和 Pascal 等版本，适合于科学研究者和工程技术人员直接应用。该书的程序包由 200 多个高效、实用的子程序构成，这些子程序一般有较好的数值特性，比较可靠，为各国的研究者信赖。

具有 Fortran 和 C 等高级计算机语言知识的读者可能已经注意到，如果用它们去进行程序设计，尤其当涉及矩阵运算或画图时，则编程会很麻烦。例如，若想求解一个线性代数方程，用户得首先去编写一个主程序，然后编写一个子程序去读入各个矩阵的元素，之后再编写一个子程序，求解相应的方程 (如使用 Gauss 消去法)，最后输出计算结果。如果选择的计算子程序不是很可靠，则所得的计算结果往往可能会出现问题。如果没有标准的子程序可以调用，则用户往往要将自己编好的子程序逐条地输入计算机，然后进行调试，最后进行计算。这样一个简单的问题往往需要用户编写 100 条左右的源程序，输入与调试程序也是很费事的，并无法保证所输入的程序完全可靠。求解线性方程组这样一个简单的功能需要 100 条源程序，其他复杂的功能往往要求有更多条语句，如采用双步 QR 法求取矩阵特征值的子程序则需要 500 多条源程序，其中任何一条语句有毛病，甚至调用不当 (如数组维数不匹配) 都可能导致错误结果的出现。

尽管如此，数学软件包仍在继续发展，其发展方向是采用国际上最先进的数值算法，提供更高效、更稳定、更快速、更可靠的数学软件包。例如，在线性代数计算领域，全新的 LaPACK 已经成为当前最有影响的软件包，但它们的目的似乎已经不再为一般用户提供解决问题的方法，而是为数学软件提供底层的支持。新版的 MATLAB 语言以及自由软件 Scilab 等著名的计算机数学语言都已经放弃了一直使用的 LINPACK 和 EISPACK，而采用 LaPACK 为其底层支持软件包。

一些数学的专门分支也出现了相关的数学程序库，支持 Fortran, C++ 等语言直接调用与编程。在互联网上同样有大量的 MATLAB 语言和其他计算机数学语言的数学工具箱，所以遇到典型问题的数学求解时，可以直接利用相关的工具箱来求解，因为其中大部分工具箱毕竟还是在相应领域有影响的专家编写的，得出的结果比外行自己查阅书籍、论文编写的可信度要高得多。

1.2 计算机数学语言概述

1.2.1 计算机数学语言

1984 年正式推出的 MATLAB 语言为数学问题的计算机求解，特别是控制系统的仿真和 CAD 发展起到了巨大的推动作用。1980 年前后，时任美国 New Mexico 大学计算机科学系主任的 Cleve Moler 教授认为用当时最先进的 EISPACK 和 LINPACK 软件包求解线性代数问题过程过于烦琐，所以构思一个名为 MATLAB (MATrix LABoratory，即矩阵实验室) 的交互式计算机语言。该语言 1980 年出现了免费版本，1984 年 The MathWorks 公司成立，并推出了 1.0 版。该语言的出现正赶上控制界基于状态空间的控制理论蓬勃发展的阶段，所以很快就引起了控制界学者的关注，出现了用 MATLAB 语言编写的控制系统工具箱，在控制界产生了巨大的影响，成为控制界的标准计算机语言。后来由于控制界及相关领域提出的各种各样要求，MATLAB 语言得到了持续发展，使得其功能越来越

强大。可以说，MATLAB 语言是由计算数学专家首创的，但是由控制界学者“捧红”的新型计算机语言。目前大部分工具箱都是面向控制和相关学科的，但随着 MATLAB 语言的不断发展，目前也在其他领域开始使用。稍后出现的 Mathematica 及 Maple 等语言当前也是应用广泛的计算机数学语言。此外，法国国家计算机科学与控制研究院 INRIA 开发的自由软件 Scilab 也可以部分解决常用的数学问题，其最显著的特色是完全免费且源代码全部公开，但在求解数学问题的功能上尚无法和 MATLAB 等计算机数学语言媲美。本书将在附录中简单介绍该语言的使用方法。

1.2.2 3 个代表性计算机数学语言

目前在国际上有 3 个计算机数学语言最有影响：The MathWorks 公司的 MATLAB 语言、Wolfram Research 公司的 Mathematica 语言和 Waterloo Maple 公司的 Maple 语言。这 3 个语言各有特色，其中 MATLAB 长于数值运算，其程序结构类似于其他计算机语言，因而编程很方便。Mathematica 和 Maple 有强大的解析运算和数学公式推导、定理证明的功能，相应的数值计算能力比 MATLAB 要弱，这两个语言更适合于纯数学领域的计算机求解。

和 Mathematica 及 Maple 相比，MATLAB 语言的数值运算功能是很出色的。除此之外，更有一个另两种语言不可替代的优势，就是 MATLAB 语言对各种各样领域均有领域专家编写的工具箱，可以高效、可靠地解决各种各样的问题。MATLAB 的符号运算工具箱利用 Maple 作为其符号运算引擎，能直接求解常用的符号运算问题。另外，MATLAB 提供了对 Maple 全部函数的接口，无需安装 Maple 就可以调用 Maple 所有的数学函数，这大大地增强了 MATLAB 的符号运算功能，在这方面的功能也不逊色于 Mathematica 和 Maple。故本书采用 MATLAB 语言为主要计算机数学语言，系统介绍其在数学问题求解中的应用。掌握了该语言将提高读者求解数学问题的能力，提高数学水平，拓广知识面，使得原来看起来无从下手的高深应用数学问题的实际求解变得轻而易举。

1.3 关于本书及相关内容

本书相应的课程是一门新型的课程。此前一些高校陆续开出了相应课程，如“数学实验”[13]课程简略介绍了计算机数学语言在一些应用数学分支中的最基本的分析方法，但缺乏如何利用实用的计算机数学语言系统、深入地与各个数学分支的数学问题求解有机结合。另一门相关的课程“MATLAB 语言及应用”更侧重于 MATLAB 语言的编程内容，对数学问题求解介绍也不全面。依照作者本人多年一线教学经验看，如果能找出一种中间途径，既介绍 MATLAB 编程的基本方法，更能全面系统地介绍其在应用数学各个分支的问题求解中的应用，无疑将会对读者大有裨益，这就是编写本书的初衷。

1.3.1 本书框架设计及内容安排

本书各章的安排如下：

第 1 章 (本章)，综述 MATLAB 等计算机数学语言的发展概况，介绍学习本课程的必要性及本课程与其他课程之间的关系。

第 2 章 “MATLAB 语言程序设计基础”，以比较简洁的形式对 MATLAB 语言编程、科学绘图等方面进行介绍，为学习本课程打下必要的基础。

第 3 章 “微积分问题的计算机求解”，包括微积分问题的解析解、数值微分、数值积分，其解析解部分基本涵盖了高等数学课程的全部计算内容。

第 4 章 “线性代数问题的计算机求解”，包括矩阵基本分析、矩阵基本变换、线性方程组的计算机求解、矩阵函数的求解等。

第 5 章 “积分变换与复变函数问题的计算机求解”，包括 Laplace 变换、Z 变换、Fourier 变换及反变换问题的计算机推导、留数、部分分式展开。

第 6 章 “代数方程与最优化问题的计算机求解”，包括方程的解析解与数值解、无约束最优化、有约束最优化、整数规划等内容。

第 7 章 “微分方程问题的计算机求解”，包括微分方程的解析解法、常微分方程数值解概述、常微分方程组初值问题的 MATLAB 求解、特殊微分方程的求解、边值问题的求解、偏微分方程求解入门。

第 8 章 “数据插值、函数逼近问题的计算机求解”，包括插值与数据拟合、样条插值函数及基于样条插值的数值微积分运算、曲线拟合与平滑、数字信号处理与快速 Fourier 变换技术等、滤波技术与滤波器设计。

第 9 章 “概率论与数理统计问题的计算机求解”，包括概率分布与随机数生成、统计量分析、数理统计方法、统计假设检验、方差分析等。

第 10 章 “数学问题的非传统解法”，为选学内容，包括模糊逻辑与模糊推理、神经网络在数据拟合中的应用、遗传算法在最优化求解中的应用、小波理论在数据处理中的应用、粗糙集理论与应用及分数阶微积分理论与计算等。

本书还在附录中简要介绍自由软件 Scilab 及其在数学问题求解中的应用。

本书内容看似在介绍数学，但最终目的是期望读者在理解相关数学领域最基本概念的前提下，绕开纯数学，直接由计算机数学语言得出数学问题的解。所以学习本课程将使读者提高数学素养，掌握解决实际数学问题的方法，为下一步学习其他课程也打下一个较好的基础。

1.3.2 本课程与其他相关课程的关系

本书对应的课程不是数值分析类课程的 MATLAB 版本介绍，应该理解成是从更高层次，采用更有效的方法，解决实际中可能遇到的数学问题的方法论。数值分析更侧重于介绍成型的算法，侧重于介绍原始的、能充分显示问题来龙去脉的算法，在实际问题求解中这些方法是不实用的，甚至是根本不使用的。考虑求解常微分方程初值问题的求解中，数值分析课程最侧重介绍的是四阶定步长 Runge-Kutta 算法，但从实际求解的实践来看，采用定步长算法是有问题的。其一是由于算法不如变步长算法高效，有时在求解中可能花费难于接受的时间。另一个方面，也是最重要的，定步长算法在求解过程中对解的正确性没有检测环节，在求解过程中出现误差也无从知晓，故得出的结果可靠性存

在问题，而采用变步长算法能根据误差自动选择计算步长，保证求解的正确性。另外很多内容，如延迟微分方程、微分代数方程等的求解在传统数值分析课程也是不介绍的。

高等数学和各类应用数学的计算问题均可以由介绍的方法直接求解。但这并不认为高等数学类课程的理论不重要。读者可以在学好高等数学、应用数学理论的基础上更好地理解问题，更容易地解决问题。

本书对各种数学问题一般采用 3 种形式进行描述，一种是用纯数学方式描述数学问题，一种是简要介绍求解数学问题的算法，一种是解决该类数学问题的 MATLAB 语句或语句组。前两者为支持后者的背景材料，对于数学基础不是特别好的读者可以在理解数学问题描述的基础上直接学习第 3 种方法。本书重点侧重于用 MATLAB 语言直接求解数学问题的方法论，而不是该问题的数学理论。

1.4 本章要点简介

- 本章通过一些看起来用先修课程知识难以解决的数学问题求解来介绍学习计算机数学语言的重要性，并对当前国际上最好的计算机数学语言给出综述，解释了本课程选择 MATLAB 语言求解数学问题的原因。
- 本章还回顾了数学软件包和计算机数学语言的发展过程，入门性地介绍了数学问题的解析解、数值解的基本概念，并举例说明了什么时候应该使用解析解，什么时候应该使用数值解。
- 本章还介绍了本课程的框架以及本课程与其他相关课程之间的关系。

1.5 习 题

1 在你的机器上安装 MATLAB 语言环境，并键入 **demo** 命令，由给出的菜单系统和对话框原型演示程序，领略 MATLAB 语言在求解数学问题方面的能力与方法。

2 作者用 MATLAB 语言编写了给出例子的源程序，读者可以自己用 **type** 语句阅读一下源程序，对照数学问题初步理解语句的含义，编写的源程序说明由下表列出。

序号	文件名	程 序 说 明
例 1-1	clex1.m	利用 MATLAB 的符号运算工具箱求解微分问题
例 1-2	clex2.m	分别利用 MATLAB 的符号运算工具箱和数值运算功能求解多项式方程，其中用数值方法得出的结果有误差
例 1-3	clex3.m	分别利用 MATLAB 的符号运算工具箱和数值运算功能计算 Hilbert 矩阵的行列式，其中用数值方法得出的结果有很大误差
例 1-4	clex4.m	令 $x_1 = y, x_2 = \dot{y}$，则可以将原来的二阶微分方程转换成一阶微分方程组，然后就可以求解微分方程的数值解。原方程是非线性微分方程，故不存在解析解。**ode45()** 函数可以求解常微分方程组，而 **dde23()** 可以求解延迟微分方程，或更直观地采用 Simulink 绘制求解框图
例 1-5	clex5.m	线性规划问题调用最优化工具箱中的 `linprog()` 函数可以立即得出结果。若想求解整数规划问题，则需要首先安装整数规划程序 **ipslv_mex()**

第 2 章　MATLAB 语言程序设计基础

MATLAB 语言是当前国际上自动控制领域的首选计算机语言，也是很多理工科专业最适合的计算机数学语言。本书以 MATLAB 语言为主要计算机语言，系统、全面地介绍在数学运算问题中 MATLAB 语言的应用。掌握该语言不但有助于更深入理解和掌握数学问题的求解思想，提高求解数学问题的能力，而且还可以充分利用该语言，在其他专业课程的学习中得到积极的帮助。

和其他程序设计语言相比，MATLAB 语言有如下的优势：

① **简洁高效性** MATLAB 程序设计语言集成度高，语句简洁，往往用 C/C++ 等程序设计语言编写的数百条语句，用 MATLAB 语言一条语句就能解决问题，其程序可靠性高、易于维护，可以大大提高解决问题的效率和水平。

② **科学运算功能** MATLAB 语言以矩阵为基本单元，可以直接用于矩阵运算。另外，最优化问题、数值微积分问题、微分方程数值解问题、数据处理问题等都能直接用 MATLAB 语言求解。

③ **绘图功能** MATLAB 语言可以用最直观的语句将实验数据或计算结果用图形的方式显示出来，并可以将以往难以显示出来的隐函数直接用曲线绘制出来。MATLAB 语言还允许用户用可视的方式编写图形用户界面，其难易程度和 Visual Basic 相仿，这使得用户可以容易地利用该语言编写通用程序。

④ **庞大的工具箱与模块集** MATLAB 是被控制界的学者“捧红”的，是控制界通用的计算机语言，在应用数学及控制领域几乎所有的研究方向均有自己的工具箱，而且由领域内知名专家编写，可信度比较高。随着 MATLAB 的日益普及，在其他工程领域也出现了工具箱，这也大大促进了 MATLAB 语言在各个领域的应用。

⑤ **强大的动态系统仿真功能** Simulink 提供的面向框图的仿真及概念性仿真功能，使得用户能容易地建立复杂系统模型，准确地对其进行仿真分析。Simulink 的概念性仿真模块集允许用户在一个框架下对含有控制环节、机械环节和电子、电机环节的机电一体化系统进行建模与仿真，这是目前其他计算机语言无法做到的。

第 2.1 节将介绍 MATLAB 语言编程的最基本内容，包括数据结构、基本语句结构和重要的冒号表达式与子矩阵提取方法。第 2.2 节将介绍 MATLAB 语言中矩阵的基本数学运算，包括代数运算、逻辑运算、比较运算及简单的数论运算函数。第 2.3 节将介绍 MATLAB 语言的基本编程结构，如循环语句结构、条件转移语句结构、开关结构和试探结构，介绍各种结构在程序设计中的应用。第 2.4 节介绍 MATLAB 语言编程中最重要的程序结构 —— M-函数的结构与程序编写技巧。第 2.5 节将介绍基于 MATLAB 语言的二维图形绘制的方法，如各种二维曲线绘制、隐函数的曲线绘制等，并将介绍图形修饰方法等。第 2.6 节将介绍三维图形的绘制方法、三维图形旋转与视角设置等。

限于本书的篇幅，本章只能介绍 MATLAB 语言最基础的入门知识。初步掌握该语言的基本功能，就能更好地理解和使用该语言研究数学问题求解的内容，还可以为其他相关后续课程的学习打下良好的基础。

2.1 MATLAB 程序设计语言基础

2.1.1 MATLAB 语言的变量与常量

MATLAB 语言变量名应该由一个字母引导，后面可以跟字母、数字、下划线等。例如，MYvar12, MY_Var12 和 MyVar12_ 均为有效的变量名，而 12MyVar 和 _MyVar12 为无效的变量名。在 MATLAB 中变量名是区分大小写的，也就是说，Abc 和 ABc 两个变量名表达的是不同的变量，在使用 MATLAB 语言编程时一定要注意。

在 MATLAB 语言中还为特定常数保留了一些名称，虽然这些常量都可以重新赋值，但建议在编程时应尽量避免对这些量重新赋值。

- eps —— 机器的浮点运算误差限。PC 机上 eps 的默认值为 2.2204×10^{-16}，若某个量的绝对值小于 eps，则可以认为这个量为 0。
- i 和 j —— 若 i 或 j 量不被改写，则它们表示纯虚数量 j。但在 MATLAB 程序编写过程中经常事先改写这两个变量的值，如在循环过程中常用这两个变量来表示循环变量，所以应该确认使用这两个变量时没有被改写。如果想恢复该变量，则可以用语句 i=sqrt(-1) 设置，即对 -1 求平方根。
- Inf —— 无穷大量 $+\infty$ 的 MATLAB 表示，也可以写成 inf。同样地，$-\infty$ 可以表示为 -Inf。在 MATLAB 程序执行时，即使遇到了以 0 为除数的运算，也不会终止程序的运行，而只给出一个“除 0”警告，并将结果赋成 Inf，这样的定义方式符合 IEEE 的标准。从数值运算编程角度看，这样的实现形式明显优于 C 这样的非专业语言。
- NaN —— 不定式 (not a number)，通常由 0/0 运算、Inf/Inf 及其他可能的运算得出。NaN 是一个很奇特的量，如 NaN 与 Inf 的乘积仍为 NaN。
- pi —— 圆周率 π 的双精度浮点表示。
- lasterr —— 存放最新一次的错误信息。此变量为字符串型，如果在本次执行过程中没出现过错误，则此变量为空字符串。
- lastwarn —— 存放最新的警告信息。若未出现过警告，则此变量为空字符串。

2.1.2 数据结构

2.1.2.1 数值型数据

强大方便的数值运算功能是 MATLAB 语言的最显著特色。为保证较高的计算精度，MATLAB 语言中最常用的数值量为双精度浮点数，占 8 个字节 (64 位)，遵从 IEEE 记数法，有 11 个指数位、53 位尾数及一个符号位，值域的近似范围为 -1.7×10^{308} 至 1.7×10^{308}，其 MATLAB 表示为 double()。考虑到一些特殊的应用，比如图像处

理，MATLAB 语言还引入了无符号的 8 位整形数据类型，其 MATLAB 表示为 `uint8()`，其值域为 0 至 255，这样可以大大地节省 MATLAB 的存储空间，提高处理速度。此外，在 MATLAB 中还可以使用其他的数据类型，如 `int8()`, `int16()`, `int32()`, `uint16()`, `uint32()` 等，每一个类型后面的数字表示其位数，其含义不难理解。

2.1.2.2 符号型

MATLAB 还定义了“符号”型变量，以区别于常规的数值型变量，可以用于公式推导和数学问题的解析解法。进行解析运算前需要首先将采用的变量申明为符号变量，这需要用 `syms` 命令来实现。该语句具体的用法为

```
syms var_list var_props
```

其中，`var_list` 给出需要申明的变量列表，可以同时申明多个变量，中间用空格分隔，而不是用逗号等分隔。如果需要，还可以进一步申明变量的类型 `var_props`，可以使用的类型为 `real`、`positive` 等。如果需要将 `a,b` 均定义为符号变量，则可以用 `syms a b` 语句声明，该命令还支持对符号变量具体形式的设定，如 `syms a real`。

符号型数值可以通过变精度算法函数 `vpa()` 以任意指定的精度显示出来。该函数的调用格式为

vpa(A)， 或 vpa(A,n)

其中，A 为需要显示的数值或矩阵，n 为指定的有效数字位数，前者以默认的十进制位数 (32 位) 显示结果。

【例 2-1】 试显示出圆周率 π 的前 300 位有效数字。

【求解】 使用符号运算工具箱中提供的 vpa() 函数可以按任意精度显示符号变量的值，故题中要求的结果可以用下面语句立即显示出来。

```
>> vpa(pi,300)
ans =
 3.141592653589793238462643383279502884197169399375105820974944592307816406286
  2089986280348253421170679821480865132823066470938446095505822317253594081284
  8111745028410270193852110555964462294895493038196442881097566593344612847564
  823378678316527120190914564856692346034861045432664821339360726024914127
```

如果不指定位数，则将得出如下的结果：

```
>> vpa(pi)
ans =
   3.1415926535897932384626433832795
```

2.1.2.3 其他数据结构

除了用于数学运算的数值数据结构外，MATLAB 还支持下面的数据结构：

① **字符串型数据** MATLAB 支持字符串变量，可以用它来存储相关的信息。和 C 语言等程序设计语言不同，MATLAB 字符串是用单引号括起来的，而不是用双引号。

② **多维数组** 三维数组是一般矩阵的直接拓展，可以这样理解，三维数组可以直接用于彩色数字图像的描述，在控制系统的分析上也可以直接用于多变量系统的表示上。在

实际编程中还可以使用维数更高的数组。

③ **单元数组** 单元数组是矩阵的直接扩展，其存储格式类似于普通的矩阵，而矩阵的每个元素不是数值，可以认为能存储任意类型的信息，这样每个元素称为“单元”(cell)，例如，$A\{i,j\}$ 可以表示单元数组 A 的第 i 行，第 j 列的内容。

④ **类与对象** MATLAB 允许用户自己编写包含各种复杂详细的变量，亦即类变量，该变量可以包含各种下级的信息，还可以重新对类定义其计算，这在控制系统描述中特别有用。例如，在 MATLAB 的控制系统工具箱中定义了传递函数类，可以用一个变量来表示整个传递函数，还重新定义了该类的运算，如加法运算可以直接求取多个模块的并联连接，乘法运算可以求取若干模块的串联。

2.1.3 MATLAB 的基本语句结构

MATLAB 的语句有两种结构。

① **直接赋值语句** 直接赋值语句的基本结构如下：

```
赋值变量 = 赋值表达式
```

这一过程把等号右边的表达式直接赋给左边的赋值变量，并返回到 MATLAB 的工作空间。如果赋值表达式后面没有分号，则将在 MATLAB 命令窗口中显示表达式的运算结果。若不想显示运算结果，则应该在赋值语句的末尾加一个分号。如果省略了赋值变量和等号，则表达式运算的结果将赋给保留变量 **ans**。所以说，保留变量 **ans** 将永远存放最近一次无赋值变量语句的运算结果。

【例 2-2】试在 MATLAB 环境中表示矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 0\end{bmatrix}$。

【求解】在 MATLAB 语言中表示一个矩阵是很容易的事，可以由下面的 MATLAB 语句将该矩阵直接输入到工作空间中。

```
>> A=[1,2,3; 4 5,6; 7,8 0]
A =
        1       2       3
        4       5       6
        7       8       0
```

其中的 >> 为 MATLAB 的提示符，由机器自动给出，在提示符下可以输入各种各样的 MATLAB 命令。矩阵的内容由方括号括起来的部分表示，在方括号中的分号表示矩阵的换行，逗号或空格表示同一行矩阵元素间的分隔。给出了上面的命令，就可以在 MATLAB 的工作空间中建立一个 $\boldsymbol{A}$ 变量了。如果不想显示中间结果，则应该在语句末尾加一个分号，如

```
>> A=[1,2,3; 4 5,6; 7,8 0]; % 不显示结果，但进行赋值。
>> A=[[A; [1 2 3]], [1;2;3;4]; % 矩阵维数动态变化
```

【例 2-3】试在 MATLAB 环境中输入复数矩阵

$$\boldsymbol{B}=\begin{bmatrix}1+9\mathrm{j} & 2+8\mathrm{j} & 3+7\mathrm{j}\\4+6\mathrm{j} & 5+5\mathrm{j} & 6+4\mathrm{j}\\7+3\mathrm{j} & 8+2\mathrm{j} & 0+\mathrm{j}\end{bmatrix}$$

【求解】 复数矩阵的输入同样也是很简单的，在 MATLAB 环境中定义了两个记号 i 和 j，可以用来直接输入复数矩阵，这样可以通过下面的 MATLAB 语句对复数矩阵直接进行赋值。

```
>> B=[1+9i,2+8i,3+7j; 4+6j 5+5i,6+4i; 7+3i,8+2j 1i]
B =
    1.0000 + 9.0000i   2.0000 + 8.0000i   3.0000 + 7.0000i
    4.0000 + 6.0000i   5.0000 + 5.0000i   6.0000 + 4.0000i
    7.0000 + 3.0000i   8.0000 + 2.0000i        0 + 1.0000i
```

② **函数调用语句** 函数调用语句的基本结构如下：

[返回变量列表]=函数名(输入变量列表)

其中，函数名的要求和变量名的要求是一致的，一般函数名应该对应在 MATLAB 路径下的一个文件。例如，函数名 my_fun 应该对应于 my_fun.m 文件。当然，还有一些函数名需对应于 MATLAB 内核中的内在 (built-in) 函数，如 inv() 函数等。

返回变量列表和输入变量列表均可以由若干个变量名组成，它们之间应该分别用逗号。返回变量还允许用空格分隔，例如 [$\boldsymbol{U}$ $\boldsymbol{S}$ $\boldsymbol{V}$]=svd($\boldsymbol{X}$)，该函数对给定的 $\boldsymbol{X}$ 矩阵进行奇异值分解，所得的结果由 $\boldsymbol{U},\boldsymbol{S},\boldsymbol{V}$ 3 个变量返回。

2.1.4 冒号表达式与子矩阵提取

冒号表达式是 MATLAB 中很有用的表达式，在向量生成、子矩阵提取等很多方面都是特别重要的。冒号表达式的原型为

$\boldsymbol{v}$=s_1:s_2:s_3

该函数将生成一个行向量 $\boldsymbol{v}$，其中 s_1 为向量的起始值，s_2 为步距，该向量将从 s_1 出发，每隔步距 s_2 取一个点，直至不超过 s_3 的最大值就可以构成一个向量。若省略 s_2，则步距取默认值 1。下面通过例子演示冒号表达式的应用。

【例 2-4】 试探不同的步距，从 $t \in [0,\pi]$ 区间取出一些点构成向量。

【求解】 先试一下步距 0.2，这样可以用下面的语句生成一个向量。

```
>> v1=0: 0.2: pi  % 注意，最终取值为 3 而不是 π
v1 =
         0   0.2000   0.4000   0.6000   0.8000   1.0000   1.2000   1.4000
    1.6000   1.8000   2.0000   2.2000   2.4000   2.6000   2.8000   3.0000
```

下面还将尝试冒号表达式不同的写法，并得出如下的结果：

```
>> v2=0: -0.1: pi  % 步距为负，显然不能生成向量，故得出空矩阵
v2 =
   Empty matrix: 1-by-0
>> v3=0:pi    % 取默认步距 1
v3 =
     0     1     2     3
```

```
>> v4=pi:-1:0   % 逆序排列构成新向量
v4 =
    3.1416    2.1416    1.1416    0.1416
```

提取子矩阵在 MATLAB 编程中是经常需要处理的事。提取子矩阵的具体方法是

B=A(v_1，v_2)

其中，$\boldsymbol{v}_1$ 向量表示子矩阵要包含的行号构成的向量，$\boldsymbol{v}_2$ 表示要包含的列号构成的向量，这样从 $\boldsymbol{A}$ 矩阵中提取有关的行和列，就可以构成子矩阵 $\boldsymbol{B}$ 了。若 $\boldsymbol{v}_1$ 为 :，则表示要提取所有的行，$\boldsymbol{v}_2$ 亦有相应的处理结果。关键词 end 表示最后一行 (或列，取决于其位置)。

【例 2-5】 下面将列出若干命令，并加以解释，但不列出结果，读者可以自己用一个矩阵测试这些语句，体会子矩阵提取的方法。

```
>> B1=A(1:2:end, :)          % 提取 A 矩阵全部奇数行、所有列
   B2=A([3,2,1],[2,3,4]) % 提取 A 矩阵 3,2,1 行、2,3,4 列构成子矩阵
   B3=A(:,end:-1:1)          % 将 A 矩阵左右翻转，即最后一列排在最前面
```

2.2 基本数学运算

2.2.1 矩阵的代数运算

如果一个矩阵 $\boldsymbol{A}$ 有 n 行、m 列元素，则称 $\boldsymbol{A}$ 矩阵为 $n\times m$ 矩阵；若 $n=m$，则矩阵 $\boldsymbol{A}$ 又称为方阵。MATLAB 语言中定义了下面各种矩阵的基本代数运算：

① **矩阵转置** 在数学公式中一般把一个矩阵的转置记作 $\boldsymbol{A}^{\mathrm{T}}$，假设 $\boldsymbol{A}$ 矩阵为一个 $n\times m$ 矩阵，则其转置矩阵 $\boldsymbol{B}$ 的元素定义为 $b_{ji}=a_{ij}$, $i=1,\cdots,n$, $j=1,\cdots,m$，故 $\boldsymbol{B}$ 为 $m\times n$ 矩阵。如果 $\boldsymbol{A}$ 矩阵含有复数元素，则对之进行转置时，其转置矩阵 $\boldsymbol{B}$ 的元素定义为 $b_{ji}=a_{ij}^*$, $i=1,\cdots,n$, $j=1,\cdots,m$，亦即首先对各个元素进行转置，然后再逐项求取其共轭复数值。这种转置方式又称为 Hermit 转置，其数学记号为 $\boldsymbol{B}=\boldsymbol{A}^*$。MATLAB 中用 **A'** 可以求出 $\boldsymbol{A}$ 矩阵的 Hermit 转置，矩阵的转置则可以由 **A.'** 求出。

② **加减法运算** 假设在 MATLAB 工作环境下有两个矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，则可以由 $\boldsymbol{C}=\boldsymbol{A}+\boldsymbol{B}$ 和 $\boldsymbol{C}=\boldsymbol{A}-\boldsymbol{B}$ 命令执行矩阵加减法。若 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵的维数相同，它会自动地将 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵的相应元素相加减，从而得出正确的结果，并赋给 $\boldsymbol{C}$ 变量。若二者之一为标量，则应该将其遍加 (减) 于另一个矩阵。在其他情况下，MATLAB 将自动地给出错误信息，提示用户两个矩阵的维数不匹配。

③ **矩阵乘法** 假设有两个矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，其中 $\boldsymbol{A}$ 的列数与 $\boldsymbol{B}$ 矩阵的行数相等，或其一为标量，则称 $\boldsymbol{A}, \boldsymbol{B}$ 矩阵是可乘的，或称 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵的维数是相容的。假设 $\boldsymbol{A}$ 为 $n\times m$ 矩阵，而 $\boldsymbol{B}$ 为 $m\times r$ 矩阵，则 $\boldsymbol{C}=\boldsymbol{A}\boldsymbol{B}$ 为 $n\times r$ 矩阵，其各个元素为 $c_{ij}=\sum_{k=1}^{m}a_{ik}b_{kj}$，其中 $i=1,2,\cdots,n$, $j=1,2,\cdots,r$。MATLAB 语言中两个矩阵的乘法由 **C=A*B** 直接求出，且这里并不需要指定 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵的维数。如果 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵的维数相容，则可以准确无误地获得乘积矩阵 $\boldsymbol{C}$；如果二者的维数不相容，则将给出错误信息，通知用户两个矩阵是不可乘的。

④ **矩阵的左除** MATLAB 中用 “\” 运算符号表示两个矩阵的左除，$\boldsymbol{A}\backslash\boldsymbol{B}$ 为方程 $\boldsymbol{AX}=\boldsymbol{B}$ 的解 $\boldsymbol{X}$，若 $\boldsymbol{A}$ 为非奇异方阵，则 $\boldsymbol{X}=\boldsymbol{A}^{-1}\boldsymbol{B}$。如果 $\boldsymbol{A}$ 矩阵不是方阵，也可以求出 $\boldsymbol{A}\backslash\boldsymbol{B}$，这时将使用最小二乘解法来求取 $\boldsymbol{AX}=\boldsymbol{B}$ 中的 $\boldsymbol{X}$ 矩阵。

⑤ **矩阵的右除** MATLAB 中定义了 “/” 符号，用于表示两个矩阵的右除，相当于求方程 $\boldsymbol{XA}=\boldsymbol{B}$ 的解。$\boldsymbol{A}$ 为非奇异方阵时 $\boldsymbol{B}/\boldsymbol{A}$ 为 $\boldsymbol{BA}^{-1}$，但在计算方法上存在差异，更精确地，有 `B/A=(A'\B')'`。

⑥ **矩阵翻转** MATLAB 提供了一些矩阵翻转处理的特殊命令，如 `B=fliplr(A)` 命令将矩阵 $\boldsymbol{A}$ 进行左右翻转再赋给 $\boldsymbol{B}$，亦即 $b_{ij}=a_{i,n+1-j}$，而 `C=flipud(A)` 命令将 $\boldsymbol{A}$ 矩阵进行上下翻转并将结果赋给 $\boldsymbol{C}$，亦即 $c_{ij}=a_{m+1-i,j}$。`D=rot90(A)` 将 $\boldsymbol{A}$ 矩阵逆时针旋转 90° 后赋给 $\boldsymbol{D}$，亦即 $d_{ij}=a_{j,n+1-i}$。

⑦ **矩阵乘方运算** 一个矩阵的乘方运算可以在数学上表述成 $\boldsymbol{A}^x$，而其前提条件要求 $\boldsymbol{A}$ 矩阵为方阵。如果 x 为正整数，则乘方表达式 $\boldsymbol{A}^x$ 的结果可以将 $\boldsymbol{A}$ 矩阵自乘 x 次得出。如果 x 为负整数，则可以将 $\boldsymbol{A}$ 矩阵自乘 $-x$ 次，然后对结果进行求逆运算就可以得出该乘方结果。如果 x 是一个分数，例如 $x=n/m$，其中 n 和 m 均为整数，则相当于将 $\boldsymbol{A}$ 矩阵自乘 n 次，然后对结果再开 m 次方。在 MATLAB 中统一表示成 `F=A^x`。

⑧ **点运算** MATLAB 中定义了一种特殊的运算，即所谓的点运算。两个矩阵之间的点运算是它们对应元素的直接运算。例如，`C=A.*B` 表示 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵的相应元素之间直接进行乘法运算，然后将结果赋给 $\boldsymbol{C}$ 矩阵，即，$c_{ij}=a_{ij}b_{ij}$。这种点乘积运算又称为 Hadamard 乘积。注意，点乘积运算要求 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵的维数相同。可以看出，这种运算和普通乘法运算是不同的。

点运算在 MATLAB 中起着很重要的作用。例如，当 $\boldsymbol{x}$ 是一个向量时，则求取数值 $[x_i^5]$ 时不能直接写成 `x^5`，而必须写成 `x.^5`。在进行矩阵的点运算时，同样要求运算的两个矩阵的维数一致，或其中一个变量为标量。其实一些特殊的函数，如 `sin()` 也是由点运算的形式进行的，因为它要对矩阵的每个元素求取正弦值。

矩阵点运算不只可以用于点乘积运算，还可以用于其他运算的场合。例如对前面给出的 $\boldsymbol{A}$ 矩阵作 `A.^A` 运算，则新矩阵的第 (i,j) 元素为 $a_{ij}^{a_{ij}}$，这样可以得出下面的结果。

```
>> A.^A
ans =
          1          4         27
        256       3125      46656
     823543   16777216          1
```

2.2.2 矩阵的逻辑运算

早期版本的 MATLAB 语言并没有定义专门的逻辑变量。在 MATLAB 语言中，如果一个数的值为 0，则可以认为它为逻辑 0，否则为逻辑 1。新版本支持逻辑变量，且上面的定义仍有效。

假设矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 均为 $n\times m$ 矩阵，则在 MATLAB 下定义了如下的逻辑运算：

① **矩阵的与运算** 在 MATLAB 下用 & 号表示矩阵的与运算。例如，***A* & *B*** 表示两个矩阵 ***A*** 和 ***B*** 的与运算。如果两个矩阵相应元素均非 0 则该结果元素的值为 1。否则，该元素为 0。

② **矩阵的或运算** 在 MATLAB 下用 | 号表示矩阵的或运算，如果两个矩阵相应元素均非 0，则该结果元素的值为 1。否则，该元素为 0。

③ **矩阵的非运算** 在 MATLAB 下用 ~ 号表示矩阵的非运算。若矩阵相应元素为 0，则结果为 1，否则为 0。

④ **矩阵的异或运算** MATLAB 下矩阵 ***A*** 和 ***B*** 的异或运算可以表示成 xor(***A***, ***B***)。若相应的两个数一个为 0，一个非 0，则结果为 0，否则为 1。

2.2.3 矩阵的比较运算

MATLAB 语言定义了各种比较关系，如 ***C***=***A*** > ***B***，当 ***A*** 和 ***B*** 矩阵满足 $a_{ij} > b_{ij}$ 时，$c_{ij} = 1$，否则 $c_{ij} = 0$。MATLAB 语言还支持等于关系，用 == 表示，大于等于关系，用 >= 关系，还支持不等于 ~= 关系，其意义是很明显的，可以直接使用。

MATLAB 还提供了一些特殊的函数，在编程中也是很实用的。其中，find() 函数可以查询出满足某关系的数组下标。例如，若想查出矩阵 ***C*** 中数值等于 1 的元素下标，则可以给出 find(***C***==1) 命令如下：

```
>> A=[1,2,3; 4 5,6; 7,8 0]; % 输入实数矩阵
   find(A>=5)'              % 找出矩阵元素大于等于 5 的下标
ans =
      3     5     6     8
```

可以看出，该函数相当于先将 ***A*** 矩阵按列构成列向量，然后再判断哪些元素大于或等于 5，返回其下标。而 find(isnan(***A***)) 函数将查出 ***A*** 变量中为 NaN 的各元素下标。还可以用下面的格式同时返回行和列坐标。

```
>> [i,j]=find(A>=5); [i,j]
ans =
     3     1
     2     2
     3     2
     2     3
```

此外，all() 和 any() 函数也是很实用的查询函数。

```
>> all(A>=5)
ans =
     0     0     0
>> any(A>=5)
   ans =
        1     1     1     1
```

前一个命令当 $\boldsymbol{A}$ 矩阵的某列元素全等于 5 时，相应元素为 1，否则为 0。而后者在某列中含有大于或等于 5 时，相应元素为 1，否则为 0。例如若想判定一个矩阵 $\boldsymbol{A}$ 是否元素均大于或等于 5，则可以简单地写成 all($\boldsymbol{A}$(:)>=5)。

2.2.4 解析结果的化简与变换

符号运算工具箱可以用于推导数学公式，但其结果往往不是最简形式，或不是用户期望的格式，所以需要对结果进行化简处理。MATLAB 中最常用的化简函数是simple() 函数，该函数尝试各种化简函数，最终得出计算机认为最简的结果。该函数的调用格式很简单，如下：

s_1=simple(s)	从各种方法中自动选择最简格式
[s_1,how]=simple(s)	化简并返回实际采用的化简方法

其中，s 为原始表达式，s_1 为化简后表达式，how 为实际采用的化简方式，为字符串变量。除了 simple() 函数外，还有其他专门的化简函数，如 collect() 函数可以合并同类项，expand() 可以展开多项式，factor() 可以进行因式分解，numden() 可以提取多项式的分子和分母，sincos() 可以进行三角函数的化简等。这些函数的信息与调用格式可以由 help 命令得出。

【例 2-6】假设已知含有因式的多项式 $P(s)=(s+3)^2(s^2+3s+2)(s^3+12s^2+48s+64)$，试用各种化简函数对之进行处理，并理解得出的变换结果。

【求解】首先应该定义符号变量 s，这样就可以表示该多项式了。有了多项式，则先尝试 MATLAB 认为的最简形式。

```
>> syms s; P=(s+3)^2*(s^2+3*s+2)*(s^3+12*s^2+48*s+64)  % P 保持原状
P =
   (s+3)^2*(s^2+3*s+2)*(s^3+12*s^2+48*s+64)
>> simple(P)           % 经过一系列化简尝试，得出计算机认为的最简形式
ans =
  (s+3)^2*(s+2)*(s+1)*(s+4)^3
>> [a,m]=simple(P)  % 返回化简方法为因式分解方法，用 factor() 函数将得同样结果
a =
   (s+3)^2*(s+2)*(s+1)*(s+4)^3
m =
   factor
>> expand(P)           % 多项式展开方法
ans =
  s^7+21*s^6+185*s^5+883*s^4+2454*s^3+3944*s^2+3360*s+1152
```

符号运算工具箱中有一个很有用的变量替换函数 subs()，其格式为

f_1=subs(f,x_1,x_1^*)　或　f_1=subs(f,$\{x_1,x_2,\cdots,x_n\}$,$\{x_1^*,x_2^*,\cdots,x_n^*\}$)

其中，f 为原表达式。该函数的目的是将其中的 x_1 替换成 x_1^*，生成新的表达式 f_1。后一种格式表示可以一次性替换多个变量。

符号运算工具箱的结果可以通过 latex() 函数转换成科学排版语言 LaTeX 能支持的字符串，可以直接插入 LaTeX 文档。

【例 2-7】 假设函数 $f(t)=\cos(at+b)+\sin(ct)\sin(dt)$，试用 LaTeX 表示该函数的 Taylor 幂级数展开，初步体会 LaTeX 排版格式。

【求解】 相关的 Taylor 幂级数展开的内容将在第 3.2 节中介绍，这里只使用相关的 taylor() 函数，直接得出展开的结果，然后用 latex() 函数将之转换成 LaTeX 的字符串。

```
>> syms a b c d t;                   % 假设这些变量均为符号变量
   f=cos(a*t+b)+sin(c*t)*sin(d*t);   % 定义给定函数 f(t)
   f1=taylor(f);                     % 调用 taylor() 函数求取 Taylor 展开
   latex(f1)                         % 将结果转换成 LaTeX 表达式，结果从略
```

由于得出的结果很冗长，不宜列出，只用 LaTeX 结果直接嵌入文档，并经适当的手工修改得

$$\cos b-at\sin b+\left(-\frac{a^2\cos b}{2}+cd\right)t^2+\frac{a^3\sin b}{6}t^3+\left(\frac{a^4\cos b}{24}-\frac{cd^3}{6}-\frac{c^3d}{6}\right)t^4-\frac{a^5\sin b}{120}t^5$$

对于非科技文献排版工具，如 Microsoft Word 等，则没有直接的转换程序。

2.2.5 基本数论运算

MATLAB 语言还提供了一组简单的数据变换和基本数论函数，如表 2-1 所示。下面将演示其中若干函数的应用。读者还可以自己选定矩阵对其他函数实际调用，观察得出的结果，以便更好地体会这些函数。

表 2-1 基本数据变换和数论函数表

函数名	调用格式	函数说明
floor()	$\boldsymbol{n}$=floor($\boldsymbol{x}$)	将 $\boldsymbol{x}$ 中元素按 $-\infty$ 方向取整，即取不足整数，得出 $\boldsymbol{n}$，数学上记作 $\boldsymbol{n}=\lfloor\boldsymbol{x}\rfloor$
ceil()	$\boldsymbol{n}$=ceil($\boldsymbol{x}$)	将 $\boldsymbol{x}$ 中元素按 $+\infty$ 方向取整，即取过剩整数，得出 $\boldsymbol{n}$
round()	$\boldsymbol{n}$=round($\boldsymbol{x}$)	将 $\boldsymbol{x}$ 中元素按最近的整数取整，亦即四舍五入，得出 $\boldsymbol{n}$
fix()	$\boldsymbol{n}$=fix($\boldsymbol{x}$)	将 $\boldsymbol{x}$ 中元素按按离 0 近的方向取整，得出 $\boldsymbol{n}$
rat()	[$\boldsymbol{n}$,$\boldsymbol{d}$]=rat($\boldsymbol{x}$)	将 $\boldsymbol{x}$ 中元素变换成最简有理数，$\boldsymbol{n}$ 和 $\boldsymbol{d}$ 分别为分子和分母矩阵
rem()	$\boldsymbol{B}$=rem($\boldsymbol{A}$,$\boldsymbol{C}$)	$\boldsymbol{A}$ 中元素对 $\boldsymbol{C}$ 中元素求模得出的余数
gcd()	k=gcd(n,m)	求取两个整数 n 和 m 的最大公约数
lcm()	k=lcm(n,m)	求取两个整数 n 和 m 的最小公倍数
factor()	factor(n)	对 n 进行质因数分解
isprime()	$\boldsymbol{v}_1$=isprime($\boldsymbol{v}$)	判定向量 $\boldsymbol{v}$ 中的各个整数值是否为质数，若是则 $\boldsymbol{v}_1$ 向量相应的值置 1，否则为 0

【例 2-8】 考虑一组数据 $-0.2765, 0.5772, 1.4597, 2.1091, 1.191, -1.6187$，试用不同的取整方法观察所得出的结果，并进一步理解取整函数。

【求解】 可以用下面的语句将数据用向量表示，调用取整函数则得出如下的结果:

```
>> A=[-0.2765,0.5772,1.4597,2.1091,1.191,-1.6187];
   floor(A)  % 向 -∞ 方向取整
ans =
    -1     0     1     2     1    -2
>> ceil(A)   % 向 +∞ 方向取整
ans =
     0     1     2     3     2    -1
>> round(A)  % 向 0 的方向取整
ans =
     0     1     1     2     1    -2
>> fix(A)    % 取最近的整数
ans =
     0     0     1     2     1    -1
```

【例 2-9】假设 3×3 的 Hilbert 矩阵可以由 ***A***=hilb(3) 定义，试对其进行有理数变换。

【求解】 用下面的语句可以进行所需变换，并得出所需结果。

```
>> A=hilb(3); [n,d]=rat(A)
n =                            d =
      1     1     1                 1     2     3
      1     1     1                 2     3     4
      1     1     1                 3     4     5
```

【例 2-10】试求 1856120 和 1483720 的最大公约数与最小公倍数，并求出所得出的最小公倍数的质因数分解。

【求解】 由于数值较大，不适合用 MATLAB 的数值形式显示，所以有必要将其转换成符号变量，并由下面的语句直接解出所需的结果。

```
>> m=sym(1856120); n=sym(1483720); [gcd(m,n), lcm(m,n)]
ans =
   [         1960, 1405082840]
```

亦即其最大公约数为 1960，最小公倍数为 1405082840。其最小公倍数的质约数分解可以由下面的语句求出。

```
>> factor(lcm(n,m))
ans =
   (2)^3*(5)*(7)^2*(757)*(947)
```

这里使用的 gcd() 和 lcm() 函数只能用于求解两个整数的相应运算。如果想求多个数值的最大公约数与最小公倍数，则可以使用这些函数的嵌套形式，如 gcd(gcd(m,n),k)。

【例 2-11】试列出 1~1000 之间的全部质数。

【求解】 用下面的语句就可以立即求出所有满足条件的质数。这里以表 2-2 的形式给出可读性更

好的结果。在实际求解过程中，用 isprime($\boldsymbol{A}$) 测出每个整数是否为质数，最后用下标提取的方式将这些质数提取出来。

```
>> A=1:1000; B=A(isprime(A))
```

表 2-2 1000 以内的质数表

2	19	47	79	109	151	191	229	269	311	353	397	439	479	523	577	617	659	709	757	811	857	907	953
3	23	53	83	113	157	193	233	271	313	359	401	443	487	541	587	619	661	719	761	821	859	911	967
5	29	59	89	127	163	197	239	277	317	367	409	449	491	547	593	631	673	727	769	823	863	919	971
7	31	61	97	131	167	199	241	281	331	373	419	457	499	557	599	641	677	733	773	827	877	929	977
11	37	67	101	137	173	211	251	283	337	379	421	461	503	563	601	643	683	739	787	829	881	937	983
13	41	71	103	139	179	223	257	293	347	383	431	463	509	569	607	647	691	743	797	839	883	941	991
17	43	73	107	149	181	227	263	307	349	389	433	467	521	571	613	653	701	751	809	853	887	947	997

2.3 MATLAB 语言的流程结构

作为一种程序设计语言，MATLAB 提供了循环语句结构、条件语句结构、开关语句结构以及与众不同的试探语句。本节中将介绍各种语句结构。

2.3.1 循环结构

循环结构可以由 for 或 while 语句引导，用 end 语句结束，在这两个语句之间的部分称为循环体。这两种语句结构的示意图分别如图 2-1 (a)、图 2-1 (b) 所示。

① for 语句的一般结构

for $i=\boldsymbol{V}$，循环结构体，end

在 for 循环结构中，$\boldsymbol{V}$ 为一个向量，循环变量 i 每次从 $\boldsymbol{V}$ 向量中取一个数值，执行一次循环体的内容，如此下去，直至执行完 $\boldsymbol{V}$ 向量中所有的分量，将自动结束循环体的执行。由此可见，这样的格式比 C 语言的相应格式灵活得多。

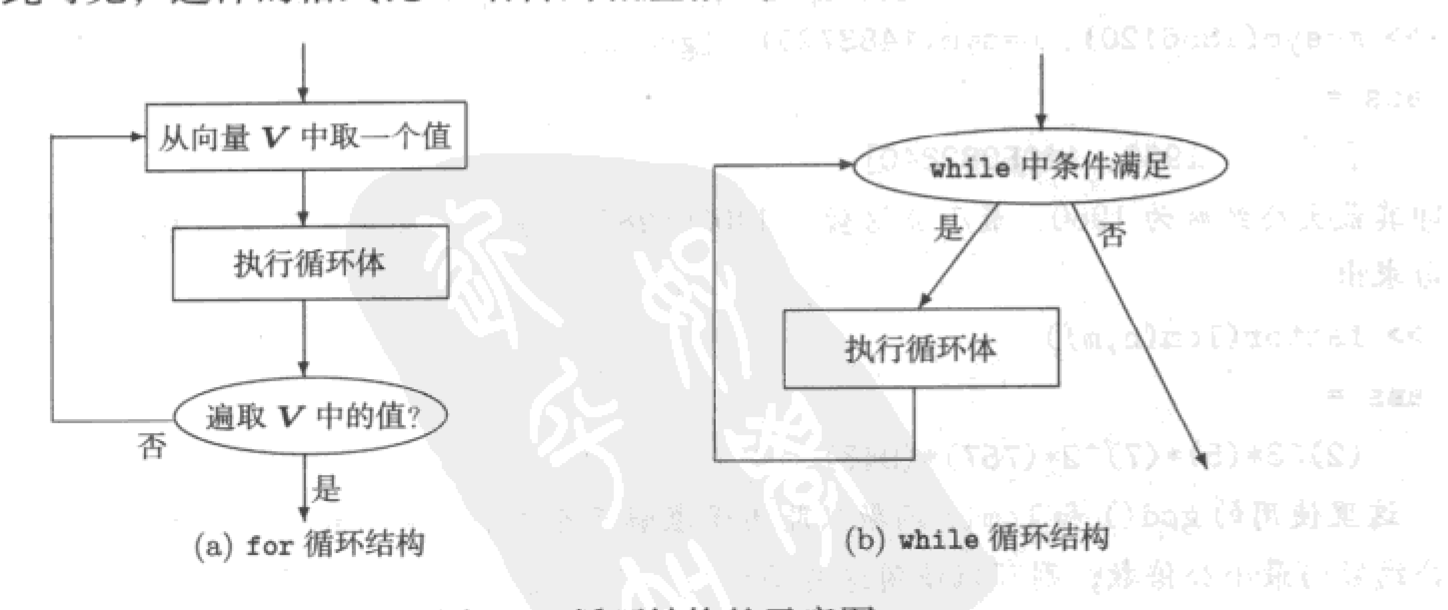


图 2-1 循环结构的示意图

② while 循环的基本结构

```
while (条件式), 循环结构体, end
```

while 循环中的“条件式”是一个逻辑表达式，若其值为真 (非零) 则将自动执行循环体的结构，执行完后再判定“条件式”的真伪，为真则仍然执行结构体，否则将退出循环结构。

while 与 for 循环是各有不同的，下面将通过例子演示它们的区别及适用场合。

【例 2-12】用循环结构求解 $\sum_{i=1}^{100} i$。

【求解】利用循环语句中的 for 结构和 while 结构，则可以按下面的语句分别编程，并得出相同的结果。

```
>> s=0;
   for i=1:100
      s=s+i;
   end
```

```
>> s=0; i=1;
   while (i<=100)
      s=s+i; i=i+1;
   end
```

其中，for 结构的编程稍简单些。事实上，前面的求和用 sum(1:100) 就能够得出所需的结果，这样做借助了 MATLAB 的 sum() 函数对整个向量进行直接操作，故程序更简单了。

【例 2-13】求出满足 $\sum_{i=1}^{m} i > 10000$ 的最小 m 值。

【求解】这样的问题用 for 循环结构就不便求解，而应该用 while 结构来求出所需的 m 值。具体的语句为

```
>> s=0; m=0;
   while (s<=10000), m=m+1; s=s+m; end, [s,m]   % 求出的 m 即是所求
ans =
     10011       141
```

循环语句在 MATLAB 语言中是可以嵌套使用的，也可以在 for 下使用 while，或相反使用。另外，在循环语句中如果使用 break 语句，则可以结束上一层的循环结构。

在 MATLAB 程序中，循环结构的执行速度较慢。所以在实际编程时，如果能对整个矩阵进行运算时，尽量不要采用循环结构，这样可以提高代码的效率。下面将通过例子演示循环与向量化编程的区别。

【例 2-14】求解级数求和问题[①] $S = \sum_{i=1}^{100000} \left(\frac{1}{2^i} + \frac{1}{3^i} \right)$。

【求解】用循环语句和向量化方式的执行时间分别可以用 tic，toc 命令测出，可见对这个问题来说，向量化所需的时间相当于循环结构的 63.7%，故用向量化的方法可以节省时间。

```
>> tic, s=0; for i=1:100000, s=s+1/i^i+1/3^i; end; toc
Elapsed time is 0.802000 seconds.
```

[①] 这里为演示向量化，夸大了求和的项数，一般求解这样的数值问题时，求取前 30 项就足够了。MATLAB 6.5 和 7.0 版均对循环进行了加速处理，尤其是 7.0 版，对本例的执行速度和向量化编程速度很接近，本例在 6.5 版下时间比约为 1:5。

```
>> tic, i=1:100000; s=sum(1./2.^i+1./3.^i); toc
Elapsed time is 0.511000 seconds.
```

2.3.2 转移结构

转移结构是一般程序设计语言都支持的结构。MATLAB 下的最基本的转移结构是 if ⋯ end 型的，也可以和 else 语句和 elseif 语句扩展转移语句。该语句的示意图如图 2-2 所示，其一般结构为：

```
if (条件 1) % 如果条件 1 满足，则执行下面的段落 1
  语句组 1 % 这里也可以嵌套下级的 if 结构
elseif (条件 2) % 否则如果满足条件 2，则执行下面的段落 2
  语句组 2
   ⋮  ⋮  % 可以按照这样的结构设置多种转移条件
else  % 上面的条件均不满足时，执行下面的段落
  语句组 n+1
end
```

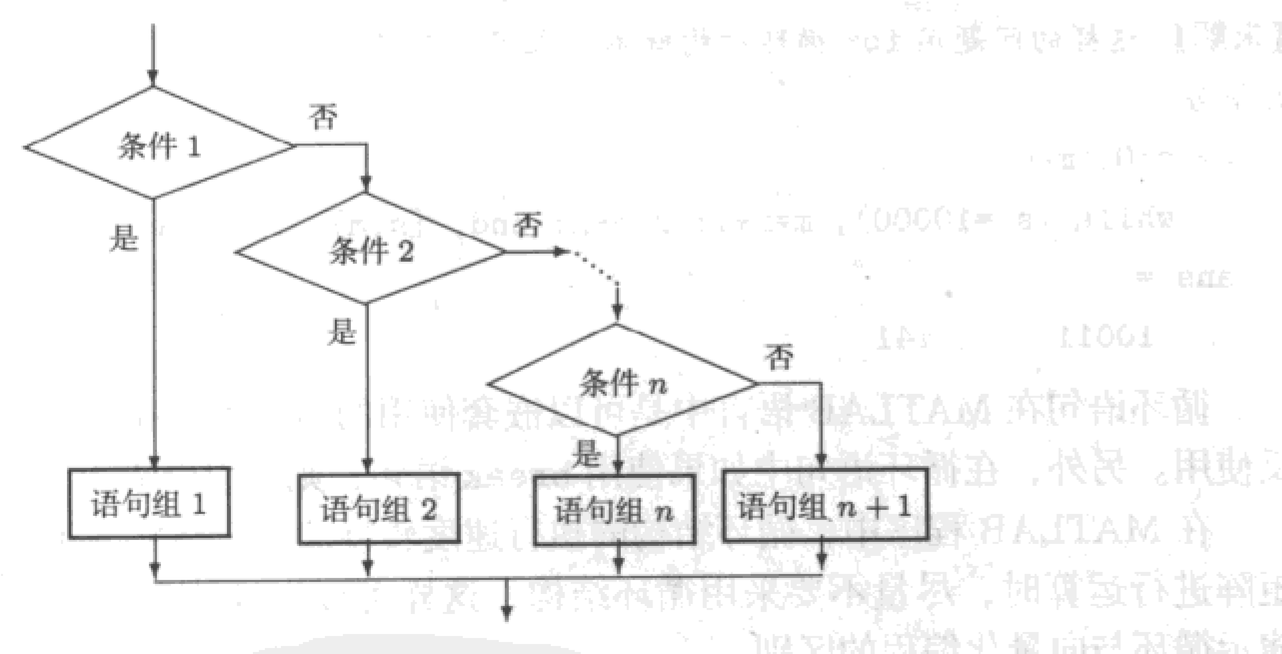


图 2-2 转移结构的示意图

【例 2-15】用 for 循环和 if 语句的形式求解例 2-13 的问题。

【求解】例 2-13 中提及只用 for 循环结构不便于实现求出和式大于 10000 的最小 i 值，利用该结构必须配合 if 语句结构才能实现。

```
s=0;
for i=1:10000
   s=s+i; if s>10000, break; end
end
```

可见，这样的结构较烦琐，不如直接使用 while 结构直观、方便。

2.3.3 开关结构

开关语句的示意图如图 2-3 所示。该语句的基本结构为：

```
switch 开关表达式
case 表达式 1
  语句段 1
case {表达式 2,表达式 3,⋯, 表达式 m}
  语句段 2
    ⋮
otherwise
  语句段 n
end
```

其中，开关语句的关键是对“开关表达式”值的判断，当开关表达式的值等于某个 case 语句后面的条件时，程序将转移到该组语句中执行，执行完成后程序转出开关体继续向下执行。

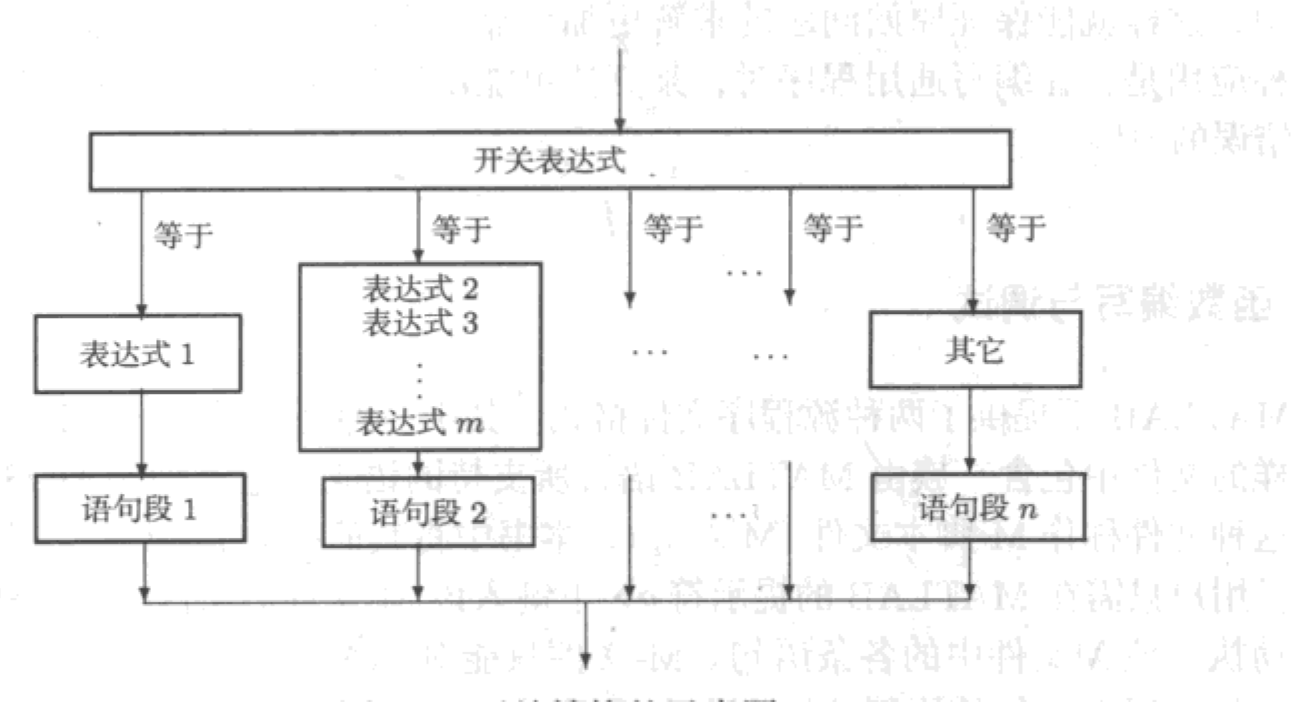


图 2-3 开关结构的示意图

在使用开关语句结构时应该注意下面几点：

① 当开关表达式的值等于表达式 1 时，将执行语句段 1，执行完语句段 1 后将转出开关体，而无需像 C 语言那样在下一个 case 语句前加 break 语句。所以本结构在这点上和 C 语言是不同的。

② 当需要在开关表达式满足若干个表达式之一时执行某一程序段，则应该把这样的一些表达式用大括号括起来，中间用逗号分隔。事实上，这样的结构是 MATLAB 语言定义的单元结构。

③ 当前面枚举的各个表达式均不满足时，则将执行 otherwise 语句后面的语句段，此语句等价于 C 语言中的 default 语句。

④ 程序的执行结果和各个 case 语句的次序是无关的。当然这也不是绝对的，当两个 case 语句中包含同样的条件，执行结果则和这两个语句的顺序有关。

⑤ 在 case 语句引导的各个表达式中，不要用重复的表达式，否则列在后面的开关通路将永远也不能执行。

2.3.4 试探结构

MATLAB 语言提供了一种新的试探式语句结构，其调用格式如下：

```
try， 语句段 1,
catch， 语句段 2,
end
```

本语句结构首先试探性地执行语句段 1，如果在此段语句执行过程中出现错误，则将错误信息赋给保留的 lasterr 变量，并终止这段语句的执行，转而执行语句段 2 中的语句。这种新的语句结构是 C 等语言中所没有的。试探性结构在实际编程中还是很实用的，例如可以将一段不保险但速度快的算法放到 try 段落中，而将一个保险的程序放到 catch 段落中，这样就能保证原始问题的求解更加可靠，且可能使程序高速执行。该结构的另外一种应用是，在编写通用程序时，某算法可能出现失效的现象，这时在 catch 语句段说明错误的原因。

2.4 函数编写与调试

MATLAB 下提供了两种源程序文件格式。其中一种是普通的 ASCII 码构成的文件，在这样的文件中包含一族由 MATLAB 语言所支持的语句，它类似于 DOS 下的批处理文件，这种文件称作 M-脚本文件 (M-script，本书中将其简称为 M-文件)，它的执行方式很简单，用户只需在 MATLAB 的提示符 >> 下键入该 M-文件的文件名，这样 MATLAB 就会自动执行该 M文件中的各条语句。M-文件只能对 MATLAB 工作空间中的数据进行处理，文件中所有语句的执行结果也完全返回到工作空间中。M-文件格式适用于用户所需要立即得到结果的小规模运算。

【例 2-16】 考虑一个实际的例子。在例 2-13 中编写一个简单的程序，可以求出和式大于 10000 的最小 m，所以若想分别求出大于 20000, 30000 的 m_i 值，分别改变程序的限制值 10000，将其设置成 20000, 30000 就可以满足要求，但这样做还是很繁杂的。如果能建立一种机制，或建立一个程序模块，给它输入 20000 的值就能返回满足它的 m_i 值，无疑这样的要求是很合理的。

在实际的 MATLAB 程序设计中，前面的一种修改程序本身的方法为 M-文件的方法，而后一种方法为函数的基本功能。后面将继续介绍函数的编写与应用。

M-函数格式是 MATLAB 程序设计的主流，在实际编程中，不建议使用 M-脚本文件格式编程。本节将着重介绍 MATLAB 函数的编写方法与技巧。

2.4.1 MATLAB 语言函数的基本结构

MATLAB 的 M-函数是由 function 语句引导的，其基本结构如下：

```
function [返回变量列表] = 函数名 (输入变量列表)
  注释说明语句段，由 % 引导
  输入、返回变量格式的检测
  函数体语句
```

这里输入和返回变量的实际个数分别由 nargin 和 nargout 两个 MATLAB 保留变量来给出，只要进入该函数，MATLAB 就将自动生成这两个变量。

返回变量如果多于 1 个，则应该用方括号将它们括起来，否则可以省去方括号。输入变量之间用逗号来分隔，返回变量用逗号或空格分隔。注释语句段的每行语句都应该由百分号 (%) 引导，百分号后面的内容不执行，只起注释作用。用户采用 help 命令则可以显示出来注释语句段的内容。此外，正规的变量个数检测也是必要的。如果输入或返回变量格式不正确，则应该给出相应的提示。

从系统的角度来说，MATLAB 函数是一个变量处理单元，它从主调函数接收变量，对之进行处理后，将结果返回到主调函数中，除了输入和输出变量外，其他在函数内部产生的所有变量都是局部变量，在函数调用结束后这些变量均将消失。这里将通过下面的例子来演示函数编程的格式与方法。

【例 2-17】先考虑例 2-16 中要求的 M-函数实现。根据要求，可以选择实际的输入变量为 k，返回的变量为 m 和 s，其中 s 为 m 项的和，这样就可以编写出该函数为

```
function [m,s]=findsum(k)
s=0; m=0;
while (s<=k), m=m+1; s=s+m; end
```

编写了函数，就可以将其存为 findm.m 文件，这样就可以在 MATLAB 环境中对不同的 k 值调用该函数了。例如，若想求出大于 145323 的最小 m 值，则可以得出如下命令：

```
>> [m1,s1]=findsum(145323)
m1 =
   539
s1 =
      145530
```

可见，这样的调用格式很灵活，无需修改程序本身就可以很容易地调用函数，得出所需的结果，所以建议采用这样的方法进行编程。

【例 2-18】假设想编写一个函数生成 $n\times m$ 阶的 Hilbert 矩阵[①]，它的第 i 行第 j 列的元素值为 $h_{i,j}=1/(i+j-1)$。想在编写的函数中实现下面几点

① 如果只给出一个输入参数，则会自动生成一个方阵，即令 $m=n$;

---

① MATLAB 中提供了生成 Hilbert 矩阵的函数 hilb()，这里只是演示函数的编写方法，而在实际使用时还是应该采用 hilb() 函数。事实上，hilb() 函数并不能生成长方 Hilbert 矩阵。

② 在函数中给出合适的帮助信息，包括基本功能、调用方式和参数说明；

③ 检测输入和返回变量的个数，如果有错误则给出错误信息。

【求解】 其实在编写程序时详细给出注释语句，养成一个好的习惯，无论对程序设计者还是对程序的维护者、使用者都是大有裨益的。根据上面的要求，可以编写一个 MATLAB 函数 myhilb()，文件名为 myhilb.m，并应该放到 MATLAB 的路径下。

```
function A=myhilb(n, m)
%MYHILB  本函数用来演示 MATLAB 语言的函数编写方法。
%   A=MYHILB(N, M) 将产生一个 N 行 M 列的 Hilbert 矩阵 A;
%   A=MYHILB(N) 将产生一个 NxN 的方 Hilbert 阵 A;
%
%See also: HILB.

% Designed by Professor Dingyu XUE, Northeastern University, PRC
%      5 April, 1995, Last modified by DYX at 30 July, 2001
if nargout>1, error('Too many output arguments.'); end
if nargin==1, m=n;   % 若给出一个输入，则生成方阵
elseif nargin==0 | nargin>2
   error('Wrong number of iutput arguments.');
end
for i=1: n
   for j=1:m
     A(i,j)=1/(i+j-1);
end, end
```

在这段程序中，由 % 引导的部分是注释语句，通常用来给出一段说明性的文字来解释程序段落的功能和变量含义等。由前面的第①点要求，首先测试输入的参数个数，如果个数为 1 (即 nargin 的值为 1)，则将矩阵的列数 m 赋成 n 的值，从而产生一个方阵。如果输入或返回变量个数不正确，则函数前面的语句将自动检测，并显示出错误信息。后面的双重 for 循环语句依据前面给出算法来生成一个 Hilbert 矩阵。

此函数的联机帮助信息可以由下面的命令获得。

```
>> help myhilb
MYHILB  本函数用来演示 MATLAB 语言的函数编写方法。
    A=MYHILB(N, M) 将产生一个 N 行 M 列的 Hilbert 矩阵 A;
    A=MYHILB(N) 将产生一个 NxN 的方 Hilbert 阵 A
See also: HILB.
```

注意，这里只显示了程序及调用方法，而没有把该函数中有关作者的信息显示出来。对照前面的函数可以立即发现，因为在作者信息的前面给出了一个空行，所以可以容易地得出结论，如果想使一段信息可以用 help 命令显示出来，则在它前面不应该加空行，即使想在 help 中显示一个空行，这个空行也应该由 % 来引导。

有了函数之后，可以采用下面的各种方法来调用它，并产生出所需的结果。

```
>> A=myhilb(3,4)  % 两个输入参数，返回长方形矩阵
A =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
>> A=myhilb(4)     % 一个输入参数，输出方阵
A =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
```

【例 2-19】MATLAB 函数是可以递归调用的，亦即在函数的内部可以调用函数自身。试用递归调用的方式编写一个求阶乘 $n!$ 的函数。

【求解】考虑求阶乘 $n!$ 的例子。由阶乘定义可见 $n! = n(n-1)!$，这样，n 的阶乘可以由 $n-1$ 的阶乘求出，而 $n-1$ 的阶乘可以由 $n-2$ 的阶乘求出，依此类推，直到计算到已知的 $1! = 0! = 1$，从而能建立起递归调用的关系。为了节省篇幅起见，这里略去了注释行段落。

```
function k=my_fact(n)
if nargin~=1, error('输入变量个数错误，只能有一个输入变量'); end
if nargout>1, error('输出变量个数过多'); end
if abs(n-floor(n))>eps | n<0 % 判定 n 是否为非负整数
   error('n 应该为非负整数');
end
if n>1       % 如果 n>1，进行递归调用
   k=n*my_fact(n-1);
elseif any([0 1]==n) % 0!=1!=1 为已知，为本函数出口
   k=1;
end
```

可以看出，该函数首先判定 n 是否为非负整数，如果不是则给出错误信息，如果是，则在 $n > 1$ 时递归调用该程序自身，若 $n = 1$ 或 0 时则直接返回 1。调用该函数则立即可以得出 11!。

```
>> my_fact(11)
ans =
    39916800
```

其实 MATLAB 提供了求取阶乘的函数 `factorial()`，其核心算法为 `prod(1:n)`，从结构上更简单、直观，速度也更快。

【例 2-20】试比较递归算法和循环算法在 Fibonacci 数列中应用的优劣。

【求解】递归算法无疑是解决一类问题的有效算法，但不宜滥用。现在考虑一个反例，考虑 Fibonacci 数列，$a_1 = a_2 = 1$，第 k 项 $(k = 3, 4, \cdots)$ 可以写成 $a_k = a_{k-1} + a_{k-2}$，这样很自然想

到使用递归调用算法编写相应的函数，该函数设置 $k=1,2$ 时出口为 1，这样函数清单如下：

```
function a=my_fibo(k)
if k==1 | k==2, a=1;
else, a=my_fibo(k-1)+my_fibo(k-2); end
```

该函数中略去了检测 k 是否为正整数的语句。如果想得到第 25 项，则需要给出如下的语句，同时测出运行该函数所运行的时间。

```
>> tic, my_fibo(25), toc
ans =
       75025
Elapsed time is 7.601000 seconds.
```

用递归算法求解该问题需时间为 7.6 秒。如果用递归方法求 $k=30$ 的时间将达到数小时的时间。现在用循环语句结构求解 $k=100$ 时的项，观察需要的时间。

```
>> tic, a=[1,1];
   for k=3:100, a(k)=a(k-1)+a(k-2); end, toc
Elapsed time is 0.020000 seconds.
```

可见，用一般循环方法所需的时间极短，就能算出来递归调用不可能的问题，所以在实际应用时应该注意不能滥用递归调用格式。

2.4.2 可变输入输出个数的处理

下面将介绍单元变量的一个重要应用 —— 如何建立起无限个输入或返回变量的函数调用格式。应该指出的是，当前很多 MATLAB 语言函数均采用本方法编写。

【例 2-21】MATLAB 提供的 conv() 函数可以用来求两个多项式的乘积。对于多个多项式的连乘，则不能直接使用此函数，而需要用该函数嵌套使用，这样在表示很多多项式连乘时相当麻烦。试编写一个 MATLAB 函数，使得它能直接处理任意多个多项式的乘积问题。

【求解】 可以用单元数据的形式来编写一个函数 convs()，专门解决多个多项式连乘的问题。

```
function a=convs(varargin)
a=1;
for i=1:length(varargin), a=conv(a,varargin{i}); end
```

这时，所有的输入变量列表由单元变量 varargin 表示。相应地，如有需要，也可以将返回变量列表用一个单元变量 varargout 表示。在这样的表示下，理论上就可以处理任意多个多项式的连乘问题了。例如可以用下面的格式调用该函数。

```
>> P=[1 2 4 0 5]; Q=[1 2]; F=[1 2 3]; D=convs(P,Q,F)
D =
     1     6    19    36    45    44    35    30
>> E=conv(conv(P,Q),F)  % 若采用 conv() 函数，则需要嵌套调用
E =
     1     6    19    36    45    44    35    30
>> G=convs(P,Q,F,[1,1],[1,3],[1,1])
```

```
G =

     1    11    56   176   376   578   678   648   527   315    90
```

2.4.3 inline 函数与匿名函数

有时为了描述某个数学函数的方便，可以用 inline() 函数来直接编写该函数，形式相当于前面介绍 M-函数，但无需编写一个真正的 MATLAB 文件，就可以描述出某种数学关系。inline() 函数的具体调用格式为

```
fun=inline('函数内容',自变量列表)
```

其中，'函数内容' 需要填写函数的具体语句，其内容应该与 function 格式的编写内容完全一致。自变量列表则可以列出类似于 function 格式下的每个自变量，且每个自变量均需要用单引号括起来。这样就可以动态定义出 inline() 函数，而无需给每个求解的内容再编写一个 MATLAB 程序了。inline() 函数在数学问题求解中，尤其是在后面将介绍的微分方程求解和最优化等求解上很有用。和 MATLAB 的 M-函数相比，这样的结构不支持结构较复杂的语句结构，只支持一个语句就能求出函数值的形式。例如，$f(x,y)=\sin(x^2+y^2)$ 可以用 f=inline('sin(x.^2+y.^2)','x','y') 直接定义。

匿名函数是 MATLAB 7.0 版提出的一种全新的函数描述形式，其描述格式类似于 inline() 函数，但比该函数更简洁，更容易使用。匿名函数的基本格式为

```
f=@(变量列表)函数内容，  例如，f=@(x,y)sin(x.^2+y.^2)
```

更重要地，该函数允许直接使用 MATLAB 工作空间中的变量。例如，若在 MATLAB 工作空间内已经定义了 a 和 b 变量，则匿名函数可以用 f=@(x,y)a*x.^2+b*y.^2 的格式定义数学关系式 $f(x,y)=ax^2+by^2$，这样无需将 a,b 作为附加参数在输入变量里表示出来，所以使得数学函数的定义更加方便。

注意，在匿名函数定义时，a,b 的值以当前 MATLAB 工作空间中的数值为主，在使用 @ 定义匿名函数后，a,b 的值再发生变化，则在函数中的值将不随着改变。

2.5 二维图形绘制

图形绘制与可视化是 MATLAB 语言的一大特色。MATLAB 中提供了一系列直观、简单的二维图形和三维图形绘制命令与函数，可以将实验结果和仿真结果用可视的形式显示出来。本节将介绍各种各样的图形绘制方法。

2.5.1 二维图形绘制基本语句

假设用户已经获得了一些实验数据。例如，已知各个时刻 $t=t_1,t_2,\cdots,t_n$ 和在这些时刻处的函数值 $y=y(t_1),y(t_2),\cdots,y(t_n)$，则可以将这些数据输入到 MATLAB 环境中，构成向量 $\boldsymbol{t}=[t_1,t_2,\cdots,t_n]$ 和 $\boldsymbol{y}=[y(t_1),y(t_2),\cdots,y(t_n)]$，如果用户想用图形的方式表示二者之间的关系，则给出

```
plot(t,y)
```

即可绘制二维图形。可以看出，该函数的调用是相当直观的。这样绘制出的“曲线”实

际上是给出各个数值点间的折线，如果这些点足够密，则看起来就是曲线了，故以后将称之为曲线。在实际应用中，plot() 函数的调用格式还可以进一步扩展：

① $\boldsymbol{t}$ 仍为向量，而 $\boldsymbol{y}$ 为矩阵，亦即

$$\boldsymbol{y}=\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{bmatrix}$$

则将在同一坐标系下绘制 m 条曲线，每一行和 $\boldsymbol{t}$ 之间的关系将绘制出一条曲线。注意，这时要求 $\boldsymbol{y}$ 矩阵的列数应该等于 $\boldsymbol{t}$ 的长度。

② $\boldsymbol{t}$ 和 $\boldsymbol{y}$ 均为矩阵，且假设 $\boldsymbol{t}$ 和 $\boldsymbol{y}$ 矩阵的行和列数均相同，则将绘制出 $\boldsymbol{t}$ 矩阵每行和 $\boldsymbol{y}$ 矩阵对应行之间关系的曲线。

③ 假设有多对这样的向量或矩阵，$(\boldsymbol{t}_1, \boldsymbol{y}_1)$，$(\boldsymbol{t}_2, \boldsymbol{y}_2)$，$\cdots$，$(\boldsymbol{t}_m, \boldsymbol{y}_m)$，则可以用下面的语句直接绘制出各自对应的曲线。

plot($\boldsymbol{t}_1, \boldsymbol{y}_1, \boldsymbol{t}_2, \boldsymbol{y}_2, \cdots, \boldsymbol{t}_m, \boldsymbol{y}_m$)

④ 曲线的性质，如线型、粗细、颜色等，还可以使用下面的命令进行指定。

plot($\boldsymbol{t}_1, \boldsymbol{y}_1$, 选项 1, $\boldsymbol{t}_2, \boldsymbol{y}_2$, 选项 2, $\cdots$, $\boldsymbol{t}_m, \boldsymbol{y}_m$, 选项 m)

其中，“选项”可以按表 2-3 中说明的形式给出，其中的选项可以进行组合。例如，若想绘制红色的点划线，且每个转折点上用五角星表示，则选项可以使用下面的组合形式 'r-.pentagram'。

表 2-3　MATLAB 绘图命令的各种选项

曲线线型		曲线颜色				标记符号			
选项	意义	选项	意义	选项	意义	选项	意义	选项	意义
'-'	实线	'b'	蓝色	'c'	蓝绿色	'*'	星号	'pentagram'	五角星
'--'	虚线	'g'	绿色	'k'	黑色	'.'	点号	'o'	圆圈
':'	点线	'm'	红紫色	'r'	红色	'x'	叉号	'square'	□
'-.'	点划线	'w'	白色	'y'	黄色	'v'	▽	'diamond'	◇
'none'	无线					'^'	△	'hexagram'	六角星
						'>'	▷	'<'	◁

绘制完二维图形后，还可以用 grid on 命令在图形上添加网格线，用 grid off 命令取消网格线；另外用 hold on 命令可以保护当前的坐标系，使得以后再使用 plot() 函数时将新的曲线叠印在原来的图上，用 hold off 则可以曲取消保护状态；用户可以使用 title() 函数在绘制的图形上添加标题，还可以用 xlabel() 和 ylabel() 函数给 x 和 y 坐标轴添加标注。

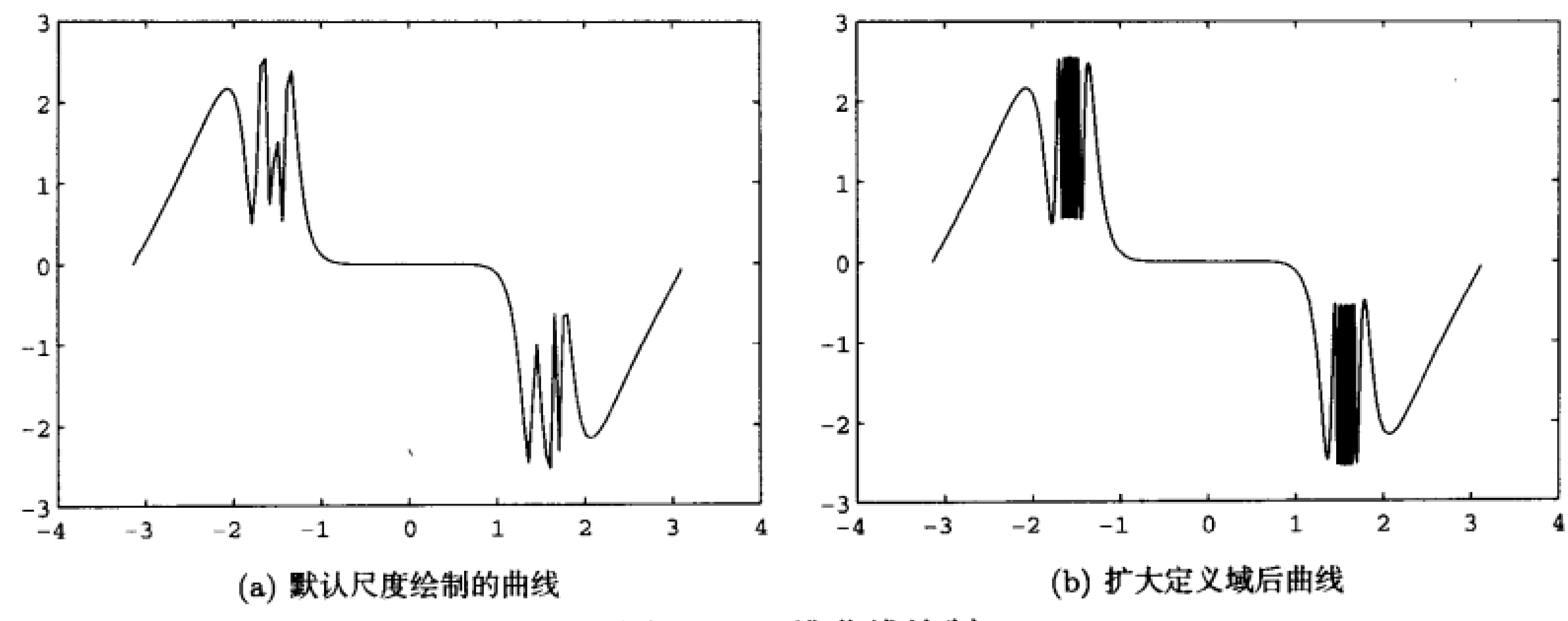
【例 2-22】试绘制出显函数方程 $y=\sin(\tan x)-\tan(\sin x)$ 在 $x\in[-\pi,\pi]$ 区间内的曲线。

【求解】解决这样问题的最直接方法可以采用下面的语句直接绘制。

```
>> x=[-pi : 0.05: pi];  % 以 0.05 为步距构造自变量向量
```

```
y=sin(tan(x))-tan(sin(x)); % 求出各个点上的函数值
plot(x,y)  % 绘制曲线
```

这些语句可以绘制出该函数的曲线，如图 2-4 (a) 所示。



(a) 默认尺度绘制的曲线　　(b) 扩大定义域后曲线

图 2-4　二维曲线绘制

从得出的曲线可以看出，在 $x \in (-1.8,-1.2)$ 及 $x \in (1.2,1.8)$ 两个子区间内图形较粗糙，应该在这些区间能加密自变量选择点，这样可以将上述的语句修改为

```
>> x=[-pi:0.05:-1.8,-1.801:.001:-1.2, -1.2:0.05:1.2,...
      1.201:0.001:1.8, 1.81:0.05:pi]; % 以变步距方式构造自变量向量
   y=sin(tan(x))-tan(sin(x)); % 求出各个点上的函数值
   plot(x,y)  % 绘制曲线
```

这样将得出如图 2-4 (b) 所示的曲线。可见，这样得出的曲线在快变化区域内表现良好。

【例 2-23】 绘制出饱和非线性特性方程 $y=\begin{cases} 1.1\mathrm{sign}(x), & |x|>1.1 \\ x, & |x|\leqslant 1.1 \end{cases}$ 的曲线。

【求解】 当然用 if 语句可以很容易求出各个 x 点上的 y 值。但这里将考虑另外一种有效的实现方法。如果构造了 x 向量，则关系表达式 x>1.1 将生成一个和 x 一样长的向量，在满足 $x_i>1.1$ 的点上，生成向量的对应值为 1，否则为 0，根据这样的想法，则可以用下面的语句绘制出分段函数的曲线，如图 2-5 所示。

```
>> x=[-2:0.02:2];  % 生成自变量向量
   y=1.1*sign(x).*(abs(x)>1.1) + x.*(abs(x)<=1.1); plot(x,y)
```

在这样的分段模型描述中，注意不要将某个区间重复表示。例如，不能将给出的语句中最后一个条件表示成 1.1*(x>=1.1)，否则因为第 2 项中也有 $x_i=1.1$ 的选项，将使得 $x_i=1.1$ 点函数求取重复，得出错误的结果。

另外，由于 plot() 函数只将给定点用直线连接起来，分段线性的非线性曲线可以由有限的几个转折点来表示，即

```
>> plot([-2,-1.1,1.1,2],[-1.1,-1.1,1.1,1.1])
```

该语句能得出和图 2-5 完全一致的结果。

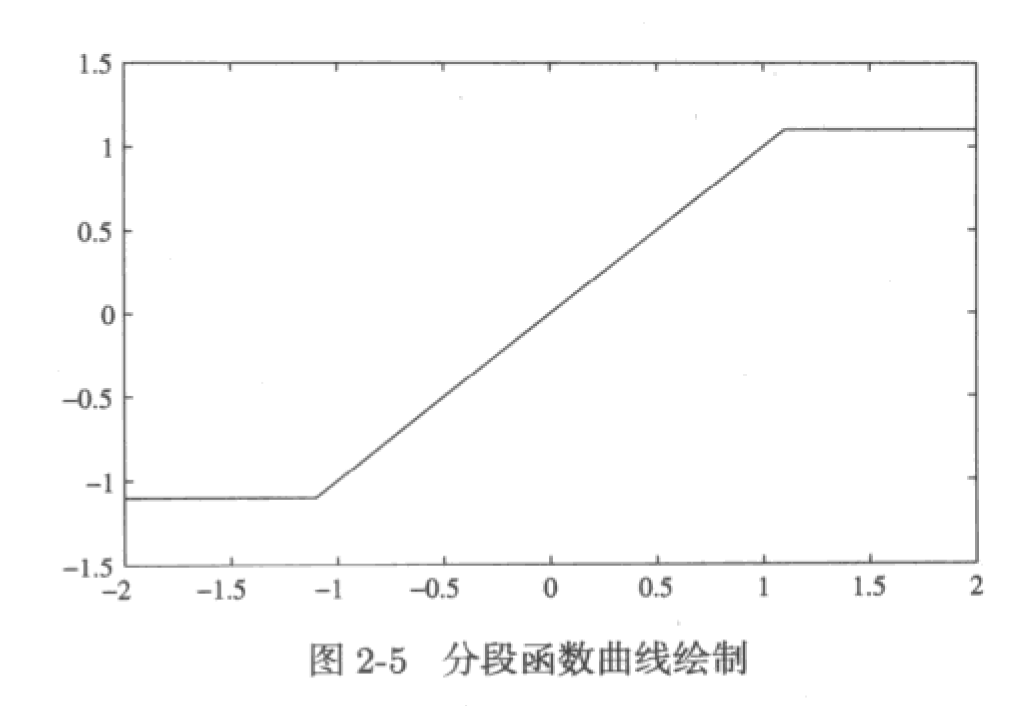


图 2-5 分段函数曲线绘制

在 MATLAB 绘制的图形中，每条曲线是一个对象，坐标轴是一个对象，而图形窗口还是一个对象，每个对象都有不同的属性，用户可以通过 set() 函数可以设置对象的属性，还可以用 get() 函数获得对象的某个属性。这两个语句的语句结构为

```
set(句柄，'属性名 1'，属性值 1，'属性名 2'，属性值 2，…)
v=get(句柄，'属性名')
```

这两个语句在界面编程中特别有用。

2.5.2 其他二维图形绘制语句

除了标准的二维曲线绘制之外，MATLAB 还提供了具有各种特殊意义的图形绘制函数，其常用调用格式如表 2-4 所示。其中，参数 $\boldsymbol{x}$，$\boldsymbol{y}$ 分别表示横、纵坐标绘图数据，c 表示颜色选项，$\boldsymbol{y}_m$，$\boldsymbol{y}_M$ 表示误差图的上下限向量。当然，随着输入参数个数及类型的不同，各个函数的绘图形式也有所区别。下面将通过例子来演示各个绘图函数的效果。

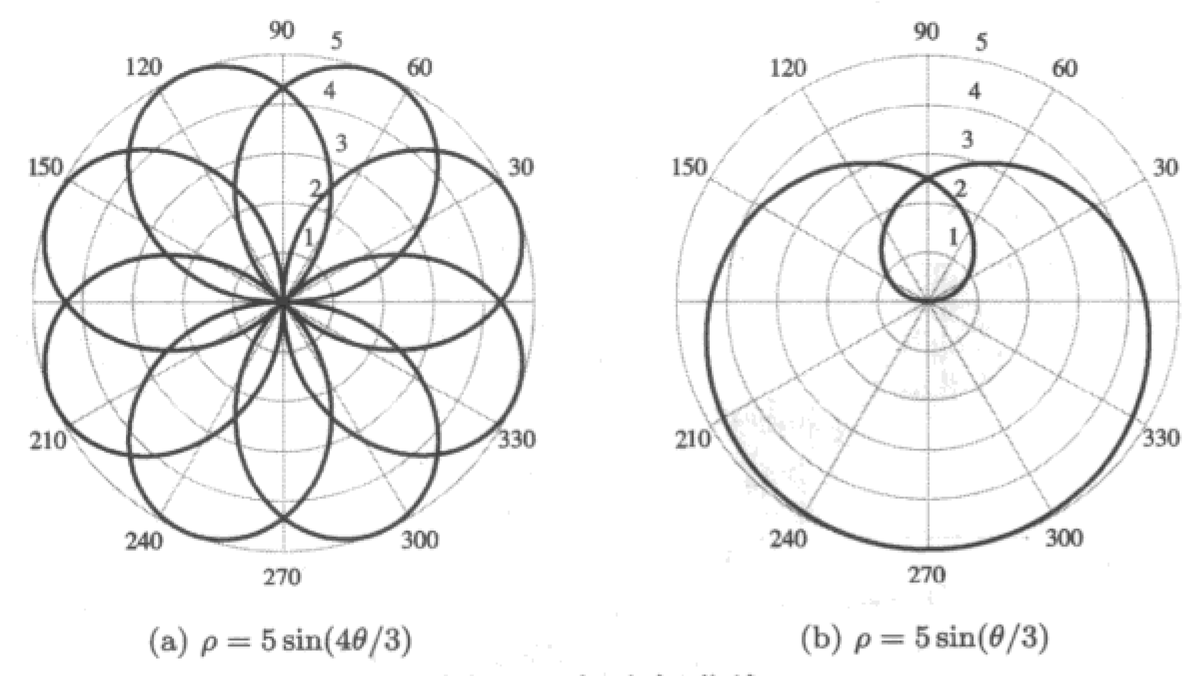
表 2-4 MATLAB 提供的特殊二维曲线绘制函数

函数名	意 义	常用调用格式	函数名	意 义	常用调用格式
bar()	二维条形图	bar($\boldsymbol{x}$,$\boldsymbol{y}$)	comet()	彗星状轨迹图	comet($\boldsymbol{x}$,$\boldsymbol{y}$)
compass()	罗盘图	compass($\boldsymbol{x}$,$\boldsymbol{y}$)	errorbar()	误差限图形	errorbar($\boldsymbol{x}$,$\boldsymbol{y}$,$\boldsymbol{y}_m$,$\boldsymbol{y}_M$)
feather()	羽毛状图	feather($\boldsymbol{x}$,$\boldsymbol{y}$)	fill()	二维填充函数	fill($\boldsymbol{x}$,$\boldsymbol{y}$,c)
hist()	直方图	hist($\boldsymbol{y}$,n)	loglog()	对数图	loglog($\boldsymbol{x}$,$\boldsymbol{y}$)
polar()	极坐标图	polar($\boldsymbol{x}$,$\boldsymbol{y}$)	quiver()	磁力线图	quiver($\boldsymbol{x}$,$\boldsymbol{y}$)
stairs()	阶梯图形	stairs($\boldsymbol{x}$,$\boldsymbol{y}$)	stem()	火柴杆图	stem($\boldsymbol{x}$,$\boldsymbol{y}$)
semilogx()	x-半对数图	semilogx($\boldsymbol{x}$,$\boldsymbol{y}$)	semilogy()	y-半对数图	semilogy($\boldsymbol{x}$,$\boldsymbol{y}$)

【例 2-24】试用极坐标绘制函数 polar() 绘制出 $\rho=5\sin(4\theta/3)$ 和 $\rho=5\sin(\theta/3)$ 的极坐标曲线。

【求解】由极坐标方程的数学表达式可以立即得出结论，这两个函数的周期均为 6π，所以若想绘

制极坐标曲线，则应该先构造一个 $\boldsymbol{\theta}$ 向量，然后求出 $\boldsymbol{\rho}$ 向量，调用 polar() 函数就可以立即绘制出所需的极坐标曲线，分别如图 2-6 (a)、(b) 所示。



(a) $\rho = 5\sin(4\theta/3)$ (b) $\rho = 5\sin(\theta/3)$

图 2-6 极坐标曲线

```
>> theta=0:0.01:6*pi; rho=5*sin(4*theta/3); polar(theta,rho)
   figure; rho=5*sin(theta/3); polar(theta,rho)
```

【例 2-25】 以正弦数据为例，试在同一窗口的不同区域用不同的绘图方式绘制出相应的曲线。

【求解】 可以用下面的各种语句绘制出如图 2-7 所示的曲线。其中，subplot() 函数可以将图形窗口分为若干块，在某一块内绘制图形。在函数调用时，第 1 个 2 表示将窗口分为 2 行，第 2 个 2 将窗口分为 2 列，第 3 个参数指定绘图的位置。

```
>> t=0:.2:2*pi; y=sin(t);          % 先生成绘图用数据
   subplot(2,2,1), stairs(t,y)    % 分割窗口，在左上角绘制阶梯曲线
   subplot(2,2,2), stem(t,y)      % 火柴杆曲线绘制
   subplot(2,2,3), bar(t,y)       % 直方图绘制
   subplot(2,2,4), semilogx(t,y)  % 横坐标为对数的曲线
```

2.5.3 隐函数绘制及应用

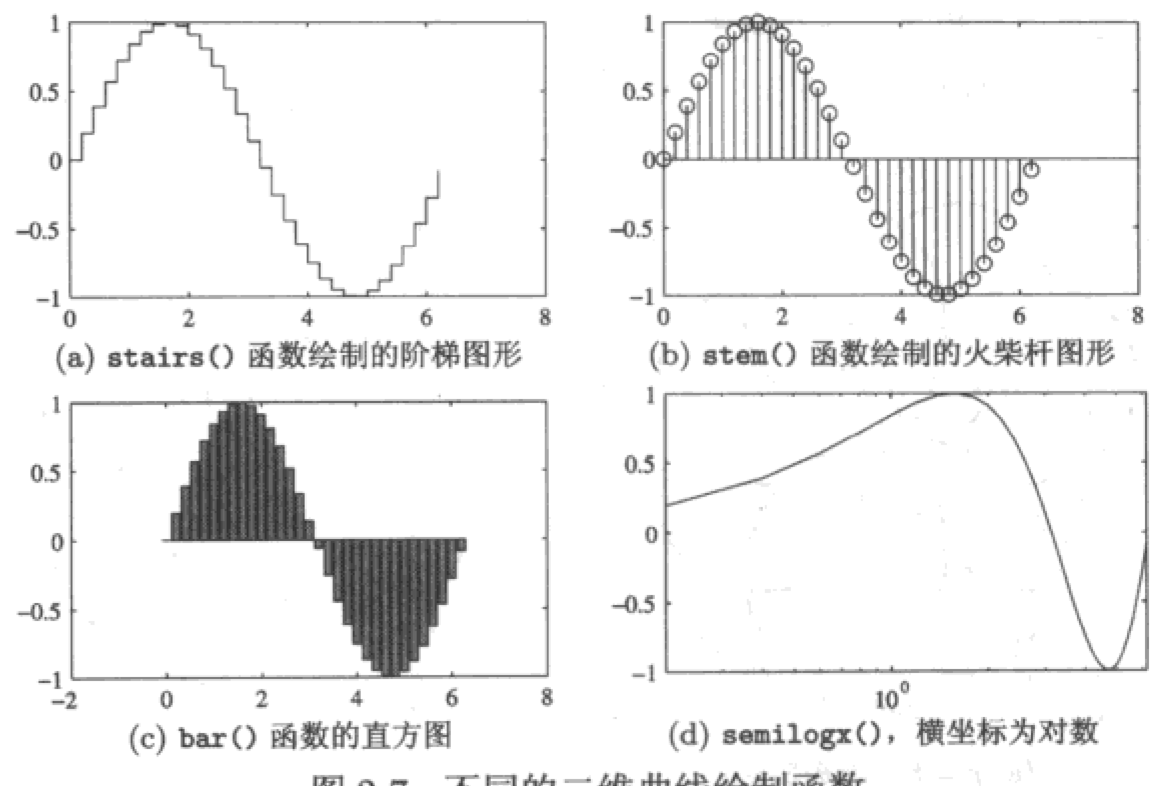
隐函数即满足 $f(x,y)=0$ 方程的 x,y 之间的关系式。用前面介绍的曲线绘制方法显然会有问题。例如，很多隐函数无法求出 x,y 之间的显式关系，所以无法先定义一个 $\boldsymbol{x}$ 向量再求出相应的 $\boldsymbol{y}$ 向量，从而不能采用 plot() 函数来绘制曲线。另外，即使能求出 x,y 之间的显式关系，但不是单值绘制，则绘制起来也是很麻烦的。MATLAB 下提供的 ezplot() 函数可以直接绘隐函数曲线，该函数的调用格式为

```
ezplot(隐函数表达式)
```

下面将通过例子来演示该函数的使用方法。

【例 2-26】 试绘制出隐函数 $f(x,y)=x^2\sin(x+y^2)+y^2\mathrm{e}^{x+y}+5\cos(x^2+y)=0$ 的曲线。

【求解】 从给出的函数可见，无法用解析的方法写出该函数，所以不能用前面给出的 plot() 函数绘制出该函数的曲线。对这样的隐函数，可以给出如下的 MATLAB 命令，并将得出如图 2-8 (a)



(a) stairs() 函数绘制的阶梯图形　(b) stem() 函数绘制的火柴杆图形

(c) bar() 函数的直方图　(d) semilogx()，横坐标为对数

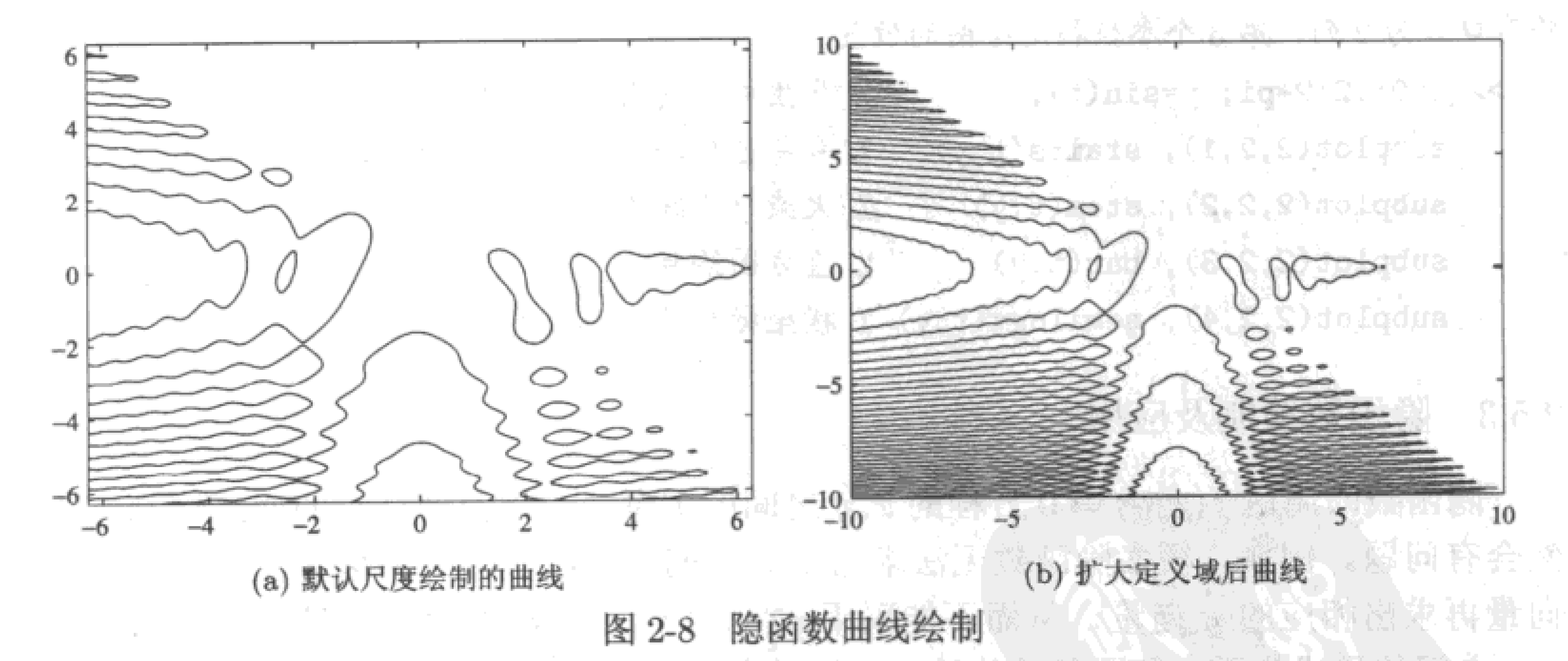
图 2-7　不同的二维曲线绘制函数

所示的隐函数曲线。

```
>> ezplot('x^2 *sin(x+y^2) +y^2*exp(x+y)+5*cos(x^2+y)')
```

上面的语句将自动选择 x 轴的范围，亦即函数的定义域，如果想改变定义域，则可以用下面的语句给出命令，并得出如图 2-8 (b) 所示的隐函数曲线。

```
>> ezplot('x^2 *sin(x+y^2) +y^2*exp(x+y)+5*cos(x^2+y)',[-10 10])
```



(a) 默认尺度绘制的曲线　(b) 扩大定义域后曲线

图 2-8　隐函数曲线绘制

2.5.4　图形修饰

MATLAB 的图形窗口工具栏中提供了各种图形修饰的功能，如在图形上添加箭头、文字及直线等，对图形的局部放大，三维图形的旋转等。图形窗口的工具栏如图 2-9 所示，如果单击“添加文字”按钮，则用鼠标在图形上单击则可以确定文字添加的位置，然后直接输入字符串即可。字符串可以用普通的字母和文字表示，也可以用 LaTeX 的格

式描述数学公式。

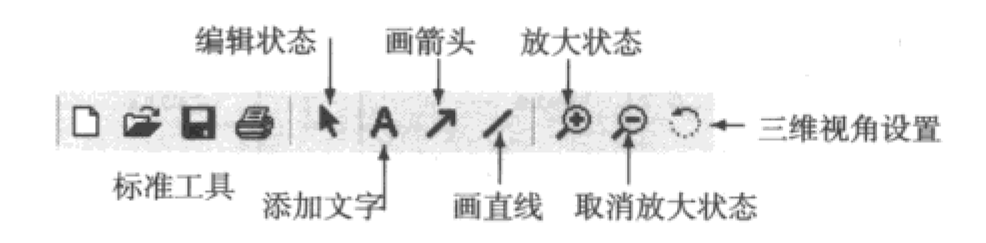


图 2-9 图形窗口工具栏

LATEX 是一个著名的科学文档排版系统，MATLAB 下支持的只是其中一个子集，这里简单介绍在 MATLAB 图形窗口中添加 LATEX 描述的数学公式的方法：

① 特殊符号是由 \ 引导的命令定义的，MATLAB 支持的特殊符号在表 2-5 中给出。

② 上下标分别用 ^ 和 _ 表示，例如 a_2^2+b_2^2=c_2^2 表示 $a_2^2+b_2^2=c_2^2$。如果需要表示多个上标，则需要用大括号括起，表示段落，例如 a^Abc 命令表示 a^Abc，其中 A 为上标。如果想将 Abc 均表示成 a 的上标，则需要给出命令 a^{Abc}。

LATEX 科技文献排版系统是当今学术界最广泛使用的排版系统，具有 Word 类排版系统无可比拟的优越性，感兴趣的读者可以进一步阅读文献 [21] 等。

MATLAB 7.0 版更新了图形修饰的功能，其编辑窗口如图 2-10 所示，其工具栏允许用户在图形上添加双向箭头、椭圆、方框等新的标记，大大提高了图形修饰的功能。

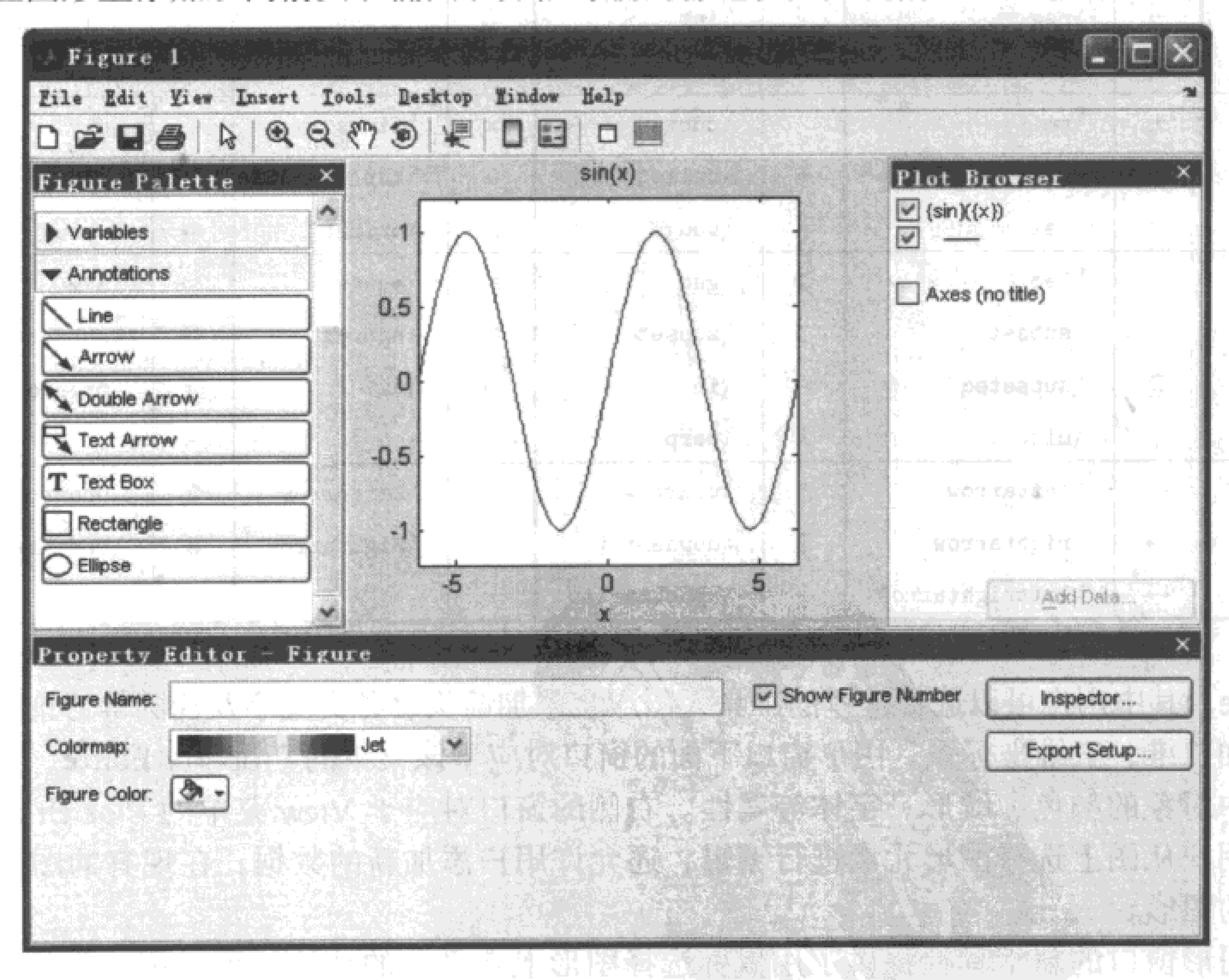


图 2-10 MATLAB 7.0 的图形编辑界面

图形编辑主要有 3 方面的内容，图形窗口左侧的部分对应于 View 菜单下的 Figure

表 2-5 图形窗口下可以直接使用的 TEX 命令表

类别	c	TEX 命令	c	TEX 命令	c	TEX 命令	c	TEX 命令
	α	\alpha	β	\beta	γ	\gamma	δ	\delta
	ϵ	\epsilon	ε	\varepsilon	ζ	\zeta	η	\eta
小写	θ	\theta	ϑ	\vartheta	ι	\iota	κ	\kappa
希腊	λ	\lambda	μ	\mu	ν	\nu	ξ	\xi
字符	o	o	π	\pi	ϖ	\varpi	ρ	\rho
	ι	\iota	κ	\kappa	ϱ	\varrho	σ	\sigma
	ς	\varsigma	τ	\tau	υ	\upsilon	ϕ	\phi
	φ	\varphi	χ	\chi	ψ	\psi	ω	\omega
大写	Γ	\Gamma	Δ	\Delta	Θ	\Theta	Λ	\Lambda
希腊	Ξ	\Xi	Π	\Pi	Σ	\Sigma	Υ	\Upsilon
字符	Φ	\Phi	Ψ	\Psi	Ω	\Omega		
	$\aleph$	\aleph	$\prime$	\prime	$\forall$	\forall	$\exists$	\exists
常用	$\wp$	\wp	$\Re$	\Re	$\Im$	\Im	∂	\partial
数学	∞	\infty	∇	\nabla	$\surd$	\surd	$\angle$	\angle
符号	$\neg$	\neg	$\int$	\int	$\clubsuit$	\clubsuit	$\diamondsuit$	\diamondsuit
	$\heartsuit$	\heartsuit	$\spadesuit$	\spadesuit				
二元	$\pm$	\pm	$\cdot$	\cdot	$\times$	\times	$\div$	\div
运算	$\circ$	\circ	$\bullet$	\bullet	$\cup$	\cup	$\cap$	\cap
符号	$\vee$	\vee	$\wedge$	\wedge	$\otimes$	\otimes	$\oplus$	\oplus
关系	$\leq$	\leq	$\geq$	\geq	$\equiv$	\equiv	$\sim$	\sim
数学	$\subset$	\subset	$\supset$	\supset	$\approx$	\approx	$\subseteq$	\subseteq
符号	$\supseteq$	\supseteq	$\in$	\in	$\ni$	\ni	$\propto$	\propto
	$\mid$	\mid	$\perp$	\perp				
箭头	$\leftarrow$	\leftarrow	$\uparrow$	\uparrow	$\Leftarrow$	\Leftarrow	$\Uparrow$	\Uparrow
符号	$\rightarrow$	\rightarrow	$\downarrow$	\downarrow	$\Rightarrow$	\Rightarrow	$\Downarrow$	\Downarrow
	$\leftrightarrow$	\leftrightarrow	$\updownarrow$	\updownarrow				

Palette，其中用户可以选择这里的工具在图形上添加箭头、各类文字及椭圆等修饰，还可以添加二维、三维坐标系。图形窗口下面的窗口对应于该菜单的 Property Editor，允许修改选中对象的颜色、线形、字体等属性。右侧的窗口对应于 View 菜单的 Plot Browser，允许用户从图上选择图形元素进行编辑，还允许用户添加新的数据，在现有的图形上叠印新的图形。

图形窗口的新工具栏提供了用鼠标选择图形上点坐标的功能，可以用其代替早期版本的 ginput() 函数，读出曲线上点坐标的信息，该功能更适合于数学问题图解方法的实现。单击工具栏的图形旋转按钮，则可以将二维图形用三维图形表示，如图 2-11 所示。

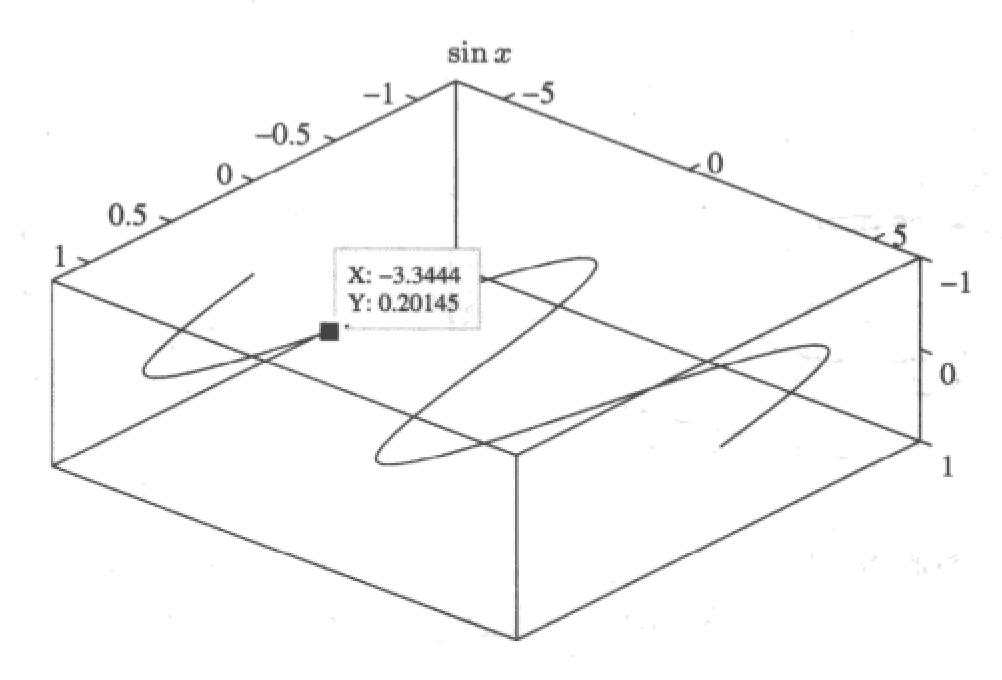


图 2-11 二维图形的三维表示

2.6 三维图形表示

2.6.1 三维曲线绘制

二维曲线绘制函数 plot() 可以扩展到三维曲线的绘制中。这时可以用 plot3() 函数绘制三维曲线。该函数的调用格式为

plot3($\boldsymbol{x}$,$\boldsymbol{y}$,$\boldsymbol{z}$)

plot3($\boldsymbol{x}_1,\boldsymbol{y}_1,\boldsymbol{z}_1$,选项 1,$\boldsymbol{x}_2,\boldsymbol{y}_2,\boldsymbol{z}_2$,选项 2,$\cdots$,$\boldsymbol{x}_m,\boldsymbol{y}_m,\boldsymbol{z}_m$,选项 m)

其中“选项”和二维曲线绘制的完全一致，如表 2-3 所示。

相应地，类似于二维曲线绘制函数，MATLAB 还提供了其他的三维曲线绘制函数，如 stem3() 可以绘制三维火柴杆型曲线，fill3() 可以绘制三维的填充图形，bar3() 可以绘制三维的直方图等。

【例 2-27】 试绘制参数方程 $x(t)=t^3\sin(3t)\mathrm{e}^{-t}, y(t)=t^3\cos(3t)\mathrm{e}^{-t}, z=t^2$ 的三维曲线。

【求解】 若想绘制该参数方程的曲线，可以先定义一个时间向量 $\boldsymbol{t}$，由其计算出 $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}$ 向量，并用函数 plot3() 绘制出三维曲线，如图 2-12 (a) 所示。注意，这里应该采用点运算。

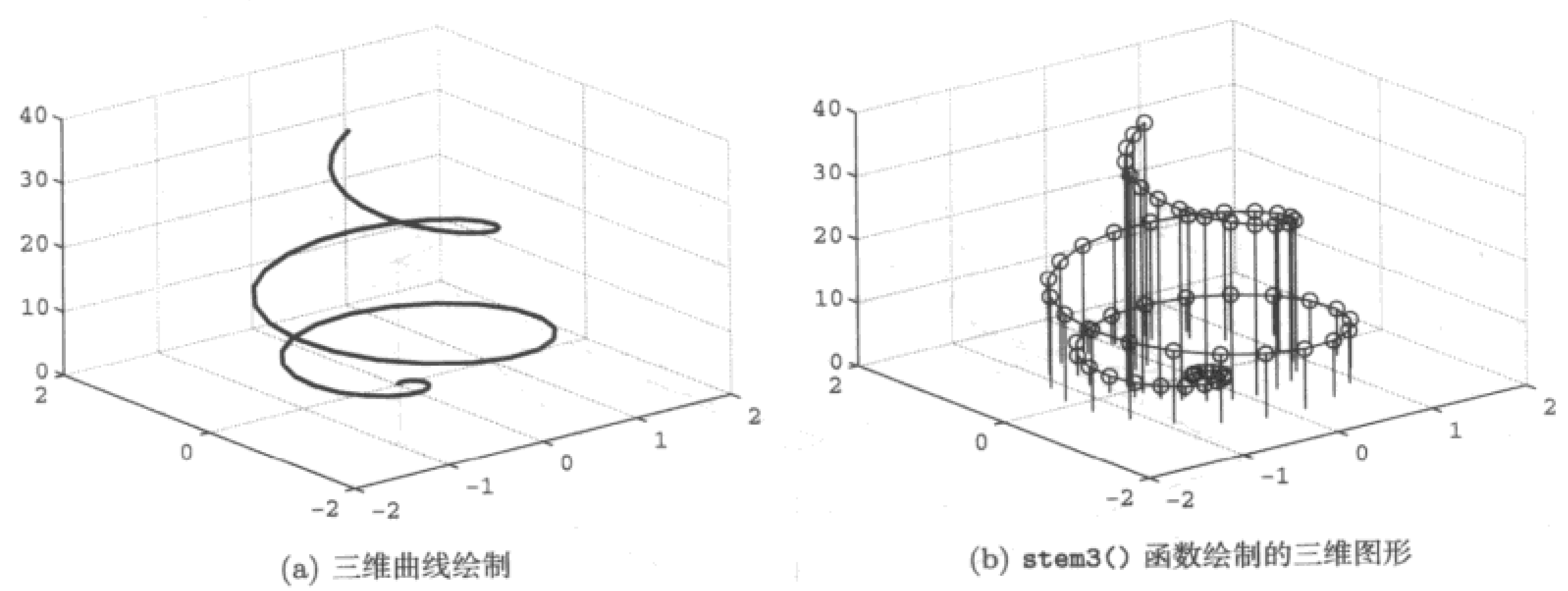
```
>> t=0:.1:2*pi;   % 构造 t 向量，注意下面的点运算
   x=t.^3.*sin(3*t).*exp(-t); y=t.^3.*cos(3*t).*exp(-t); z=t.^2;
   plot3(x,y,z), grid  % 三维曲线绘制
```

如果用 stem3() 函数绘制出火柴杆形曲线，如图 2-12 (b) 所示。

```
>> stem3(x,y,z); hold on; plot3(x,y,z), grid
```

2.6.2 三维曲面绘制

如果已知二元函数 $z=f(x,y)$，则可以绘制出该函数的三维曲面图。在绘制三维图之前，应该先调用 meshgrid() 函数生成网格矩阵数据 $\boldsymbol{x}$ 和 $\boldsymbol{y}$，这样就可以按函数公式用点运算的方式计算出 $\boldsymbol{z}$ 矩阵，之后就可以用 mesh() 或 surf() 等函数进行三维图形绘制



(a) 三维曲线绘制　　(b) stem3() 函数绘制的三维图形

图 2-12　三维曲线的绘制

了。具体的函数调用格式为

$[x,y]$=meshgrid(v_1, v_2)	生成网格数据
z= ... ，如 z=x.*y	计算二元函数的 z 矩阵
surf(x,y,z) 或 mesh(x,y,z)	mesh() 绘制网格图，surf() 绘制表面图

其中，v_1 和 v_2 为 x 和 y 轴的分隔方式。三维曲面还可以由其他函数绘制如 surfc() 函数和 surfl() 函数可以分别绘制带有等高线和光照下的三维曲面，waterfall() 函数可以绘制瀑布形三维图形。在 MATLAB 下还提供了等高线绘制的函数，如 contour() 函数和三维等高线函数 contour3()，这里将通过例子介绍三维曲面绘制方法与技巧。

【例 2-28】数字图像处理中使用的 Butterworth 低通滤波器的数学模型为[11]

$$H(u,v)=\frac{1}{1+D^{2n}(u,v)/D_0}$$

其中，$D(u,v)=\sqrt{(u-u_0)^2+(v-v_0)^2}$，$D_0$ 为给定的区域半径，n 为阶次，u_0 和 v_0 为区域的中心。假设 $D_0=200, n=2$，试用三维曲面的形式绘制该滤波器图形。

【求解】给定滤波器数学模型的三维曲面可以通过下面的语句绘制出来，如图 2-13 (a) 所示。

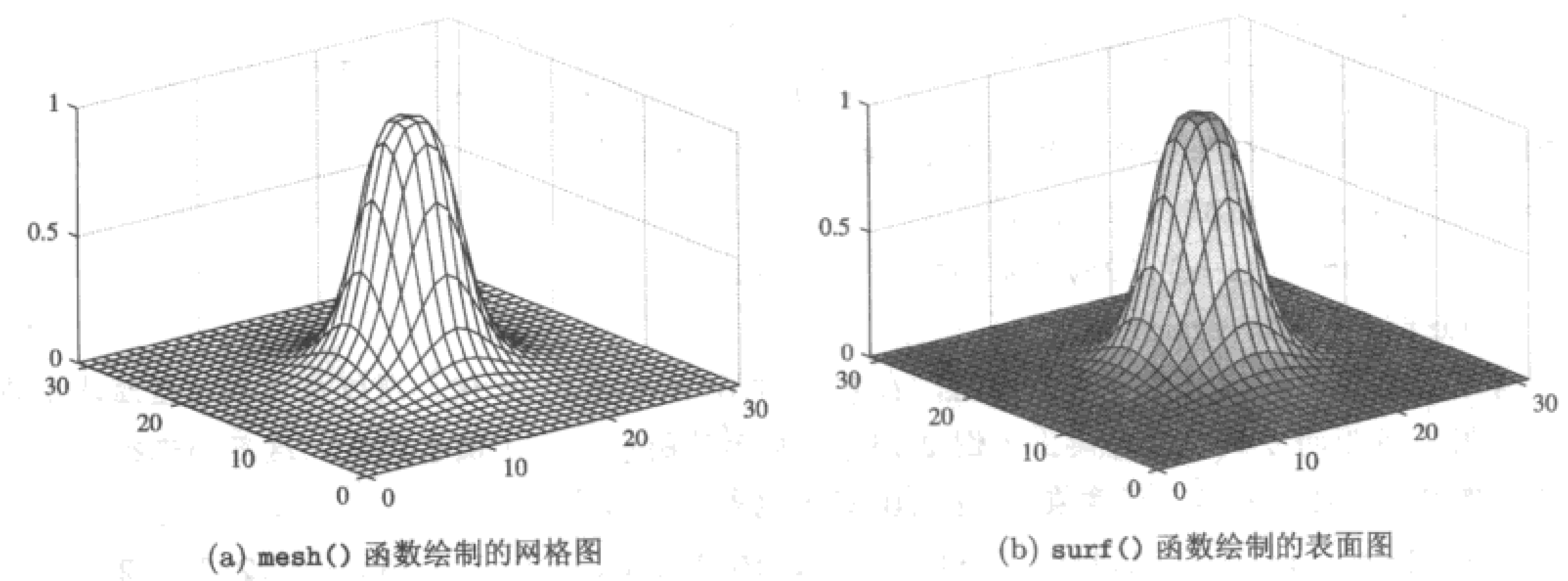
```
>> [x,y]=meshgrid(0:31);  n=2; D0=200;
   D=sqrt((x-16).^2+(y-16).^2);           % 求距离
   z=1./(1+D.^(2*n)/D0); mesh(x,y,z),    % 计算并绘制滤波器
   axis([0,31,0,31,0,1])                 % 重新设置坐标系，增大可读性
```

若用 surf() 函数取代 mesh() 函数，则可以得出如图 2-13 (b) 所示的表面图。

```
>> surf(x,y,z)    % 绘制三维表面图
```

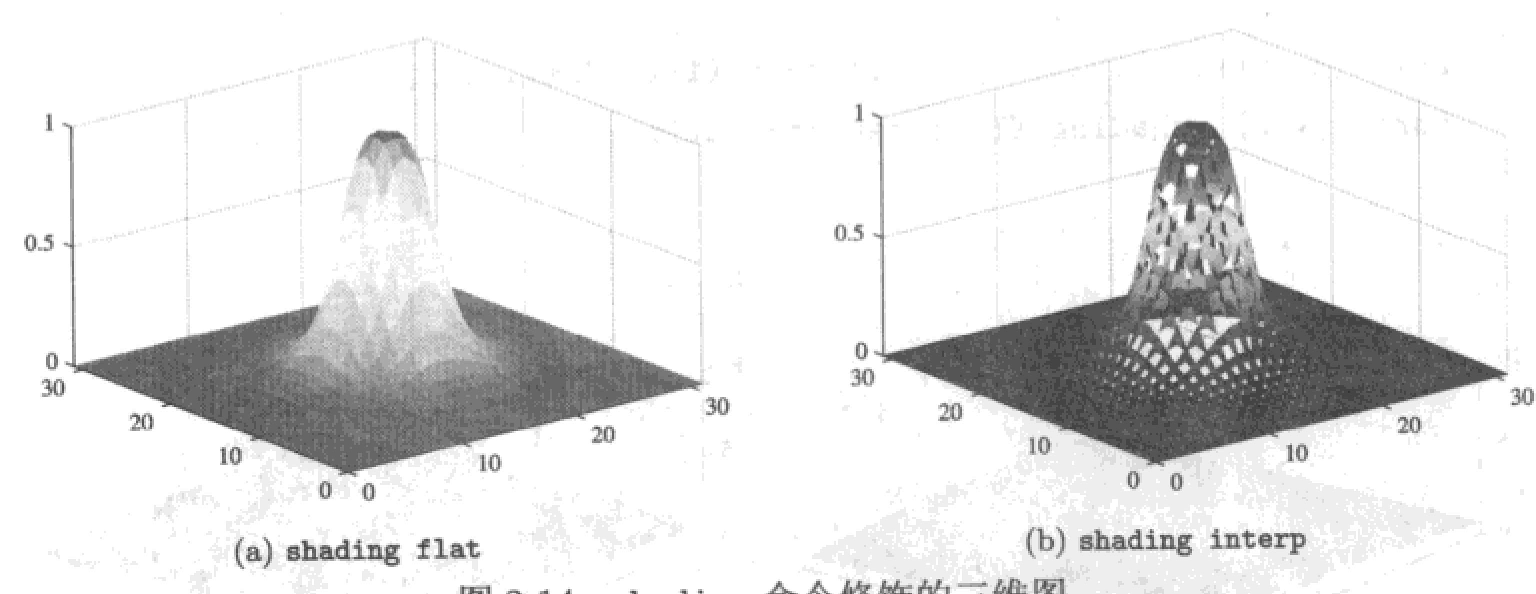
三维表面图可以用 shading 命令修饰其显式形式，该命令可以带三种不同的选项，flat (每个网格块用同样颜色着色的没有网格线的表面图，效果如图 2-14 (a) 所示)、interp (插值的光滑表面图，效果见图 2-14 (b) 所示) 和 faceted (不同于 flat，有网格线的，本选项是默认的，效果如图 2-13 (b) 所示)。

MATLAB 还提供了其他的三维图形绘制函数。如 waterfall(x,y,z) 命令可以绘制出瀑布



(a) mesh() 函数绘制的网格图　　(b) surf() 函数绘制的表面图

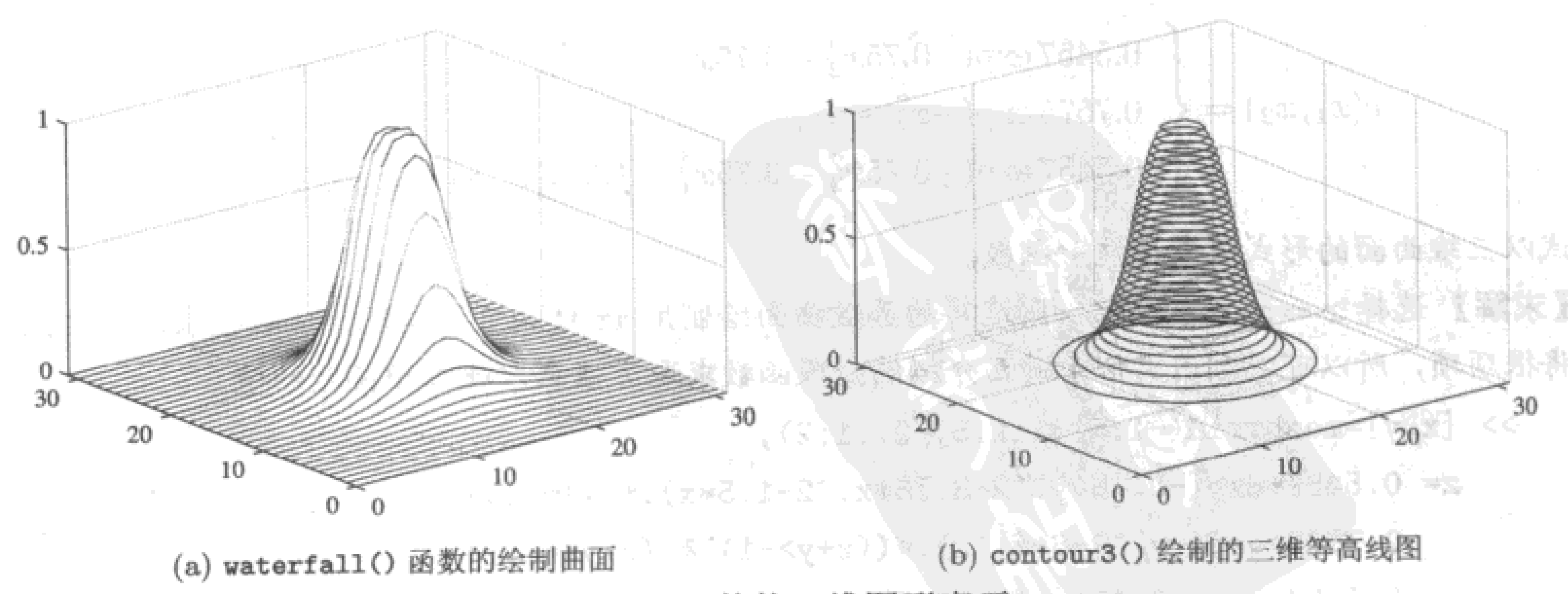
图 2-13　Butterworth 低通滤波器的三维图表示



(a) shading flat　　(b) shading interp

图 2-14　shading 命令修饰的三维图

形图形，如图 2-15 (a) 所示，而 contour3(x,y,z,30) 命令可以绘制出三维的等高线图形如图 2-15 (b) 所示。其中的 30 为用户选定的等高线条数，当然可以不给出该参数，那样将默认地设置等高线条数，对这个例子来说显得过于稀疏。



(a) waterfall() 函数的绘制曲面　　(b) contour3() 绘制的三维等高线图

图 2-15　其他三维图形表示

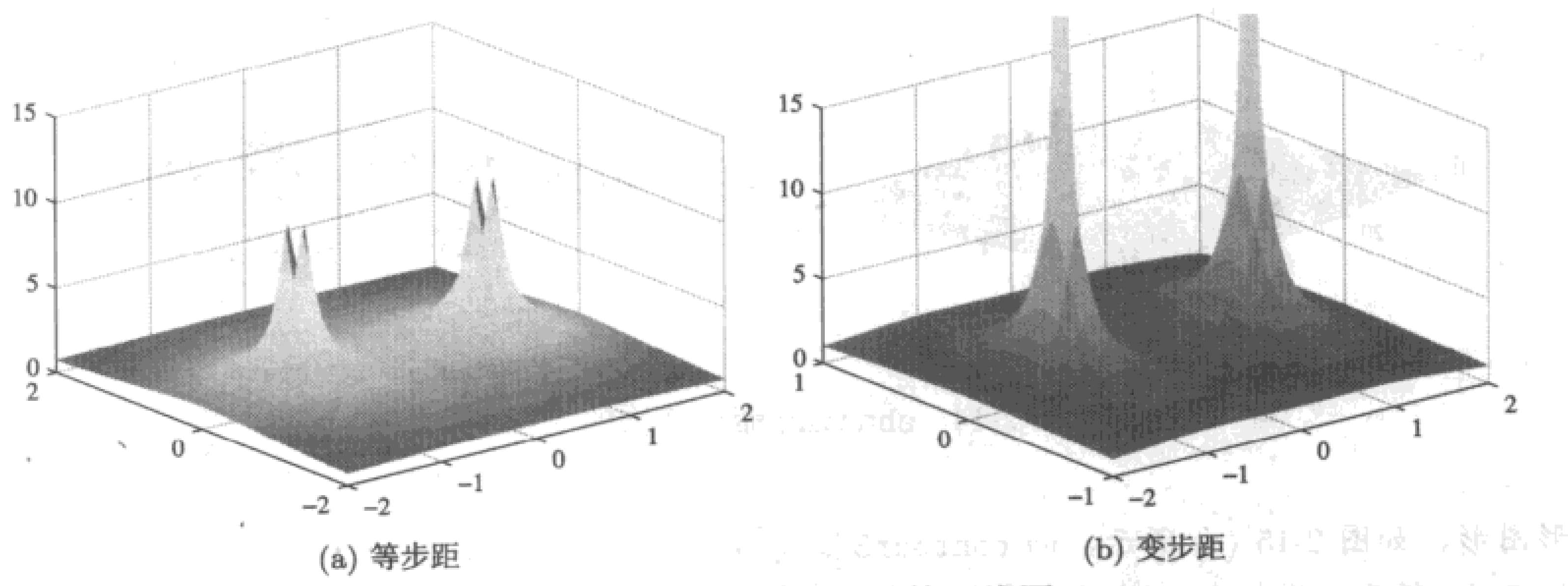
【例 2-29】 试绘制出二元函数 $z=f(x,y)=\dfrac{1}{\sqrt{(1-x)^2+y^2}}+\dfrac{1}{\sqrt{(1+x)^2+y^2}}$。

【求解】 可以用下面的语句绘制出三维图，如图 2-16 (a) 所示。

```
>> [x,y]=meshgrid(-2:.1:2);
   z=1./(sqrt((1-x).^2+y.^2))+1./(sqrt((1+x).^2+y.^2));
   surf(x,y,z), shading flat
```

事实上，这样得出的图形有点问题，在 $(-\pm1,0)$ 点处出现 ∞ 值，所以应该在该区域减小步距，采用变步距的方式，最终读出如图 2-16 (b) 所示的图形，为了便于比较，这里仍选择 z-轴的范围和图 2-16 (a) 的一致。注意在 $(-\pm1,0)$ 处的值趋于无穷大。

```
>> xx=[-2:.1:-1.2, -1.1:0.02:-0.9, -0.8:0.1:0.8, 0.9:0.02:1.1, 1.2:0.1:2];
   yy=[-1:0.1:-0.2, -0.1:0.02:0.1, 0.2:.1:1];
   [x,y]=meshgrid(xx,yy);
   z=1./(sqrt((1-x).^2+y.^2))+1./(sqrt((1+x).^2+y.^2));
   surf(x,y,z), shading flat; set(gca,'zlim',[0,15])
```



(a) 等步距 (b) 变步距

图 2-16 不同网格选择下的三维图

【例 2-30】 假设某概率密度函数由下面分段函数表示[1]。

$$p(x_1,x_2)=\begin{cases}0.5457\exp(-0.75x_2^2-3.75x_1^2-1.5x_1), & x_1+x_2>1\\ 0.7575\exp(-x_2^2-6x_1^2), & -1<x_1+x_2\leqslant 1\\ 0.5457\exp(-0.75x_2^2-3.75x_1^2+1.5x_1), & x_1+x_2\leqslant -1\end{cases}$$

试以三维曲面的形式来表示这一函数。

【求解】 选择 $x=x_1$，$y=x_2$，则这样的函数曲面绘制用 if 结构可以实现该函数值求取，但结构将很烦琐，所以可以利用类似于前面介绍的分段函数求取方法来求此二维函数的值。

```
>> [x,y]=meshgrid(-1.5:.1:1.5,-2:.1:2);
   z= 0.5457*exp(-0.75*y.^2-3.75*x.^2-1.5*x).*(x+y>1)+...
      0.7575*exp(-y.^2-6*x.^2).*((x+y>-1) & (x+y<=1))+...
      0.5457*exp(-0.75*y.^2-3.75*x.^2+1.5*x).*(x+y<=-1);
   surf(x,y,z), set(gca,'xlim',[-1.5 1.5]); shading flat
```

这样将得出如图 2-17 所示的三维表面图。

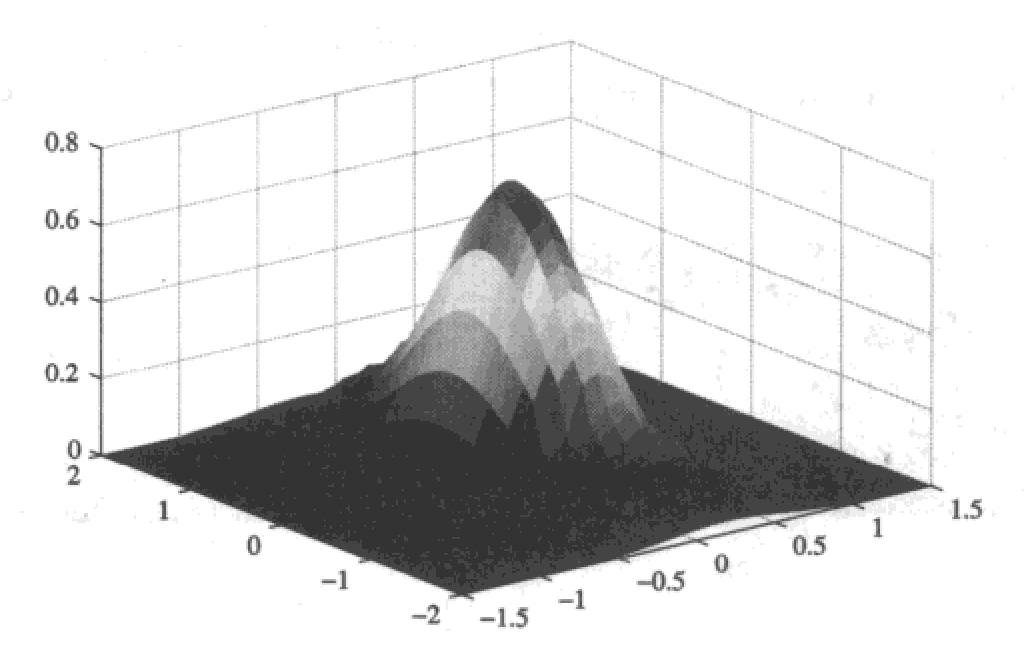
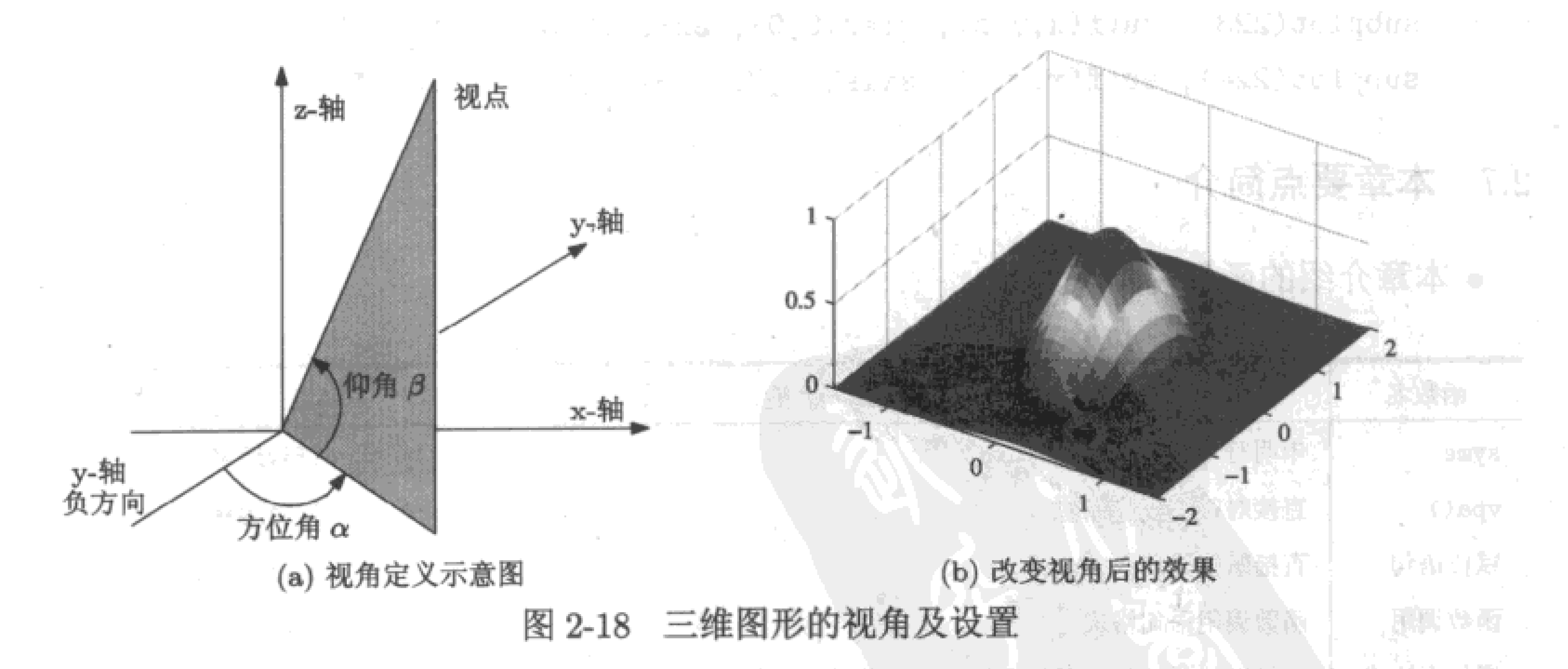


图 2-17 分段二维函数曲线绘制

2.6.3 三维图形视角设置

MATLAB 三维图形显示中提供了修改视角的功能，允许用户从任意的角度观察三维图形，实现视角转换有两种方法。其一是使用图形窗口工具栏中提供的三维图形转换按钮来可视地对图形进行旋转，其二是用 `view()` 函数有目的地进行旋转。

MATLAB 三维图形视角的定义如图 2-18 (a) 所示。其中有两个角度就可以惟一地确定视角，方位角 α 定义为视点在 x-y 平面投影点与 y 轴负方向之间的夹角，默认值为 $\alpha=-37.5°$，仰角 β 定义为视点和 x-y 平面的夹角，默认值为 $\beta=30°$。



(a) 视角定义示意图 (b) 改变视角后的效果

图 2-18 三维图形的视角及设置

如果想改变视角来观察曲面，则可以给出 `view(α,β)` 命令。例如，俯视图可以由 `view(0,90)` 设置，正视图由 `view(0,0)` 设置，侧视图可以由 `view(90,0)` 来设定。

例如，对图 2-17 中给出的三维网格图进行处理，设方位角为 $\alpha=80°$，仰角为 $\beta=10°$，则下面的 MATLAB 语句将得出如图 2-18 (b) 所示的三维曲面。

```
>> view(80,10), set(gca,'xlim',[-1.5 1.5])
```

【例 2-31】 试在同一图形窗口上绘制例 2-28 中函数曲面的三视图。

【求解】 用下面的语句可以容易地绘制出三维图，并用相应的语句设置不同的视角，则可以最终得出如图 2-19 所示的各个视图。

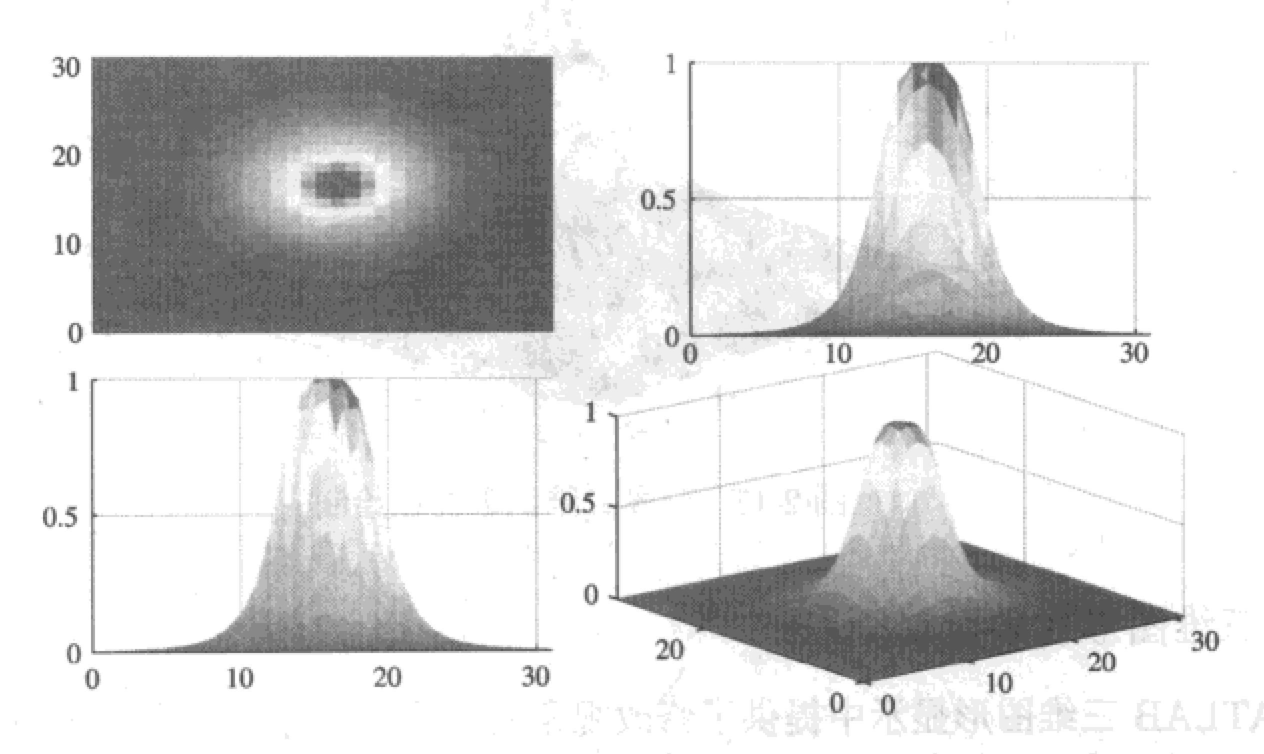


图 2-19 二元函数的三视图

```
>> [x,y]=meshgrid(0:31); n=2; D0=200;
   D=sqrt((x-16).^2+(y-16).^2); z=1./(1+D.^(2*n)/D0); % 计算滤波器
   subplot(221), surf(x,y,z), view(0,90); axis([0,31,0,31,0,1]); % 俯视图
   subplot(222), surf(x,y,z), view(90,0); axis([0,31,0,31,0,1]); % 侧视图
   subplot(223), surf(x,y,z), view(0,0); axis([0,31,0,31,0,1]);  % 正视图
   subplot(224), surf(x,y,z), axis([0,31,0,31,0,1]); % 三维图
```

2.7 本章要点简介

- 本章介绍的函数由下表给出。

函数名	函 数 功 能	工具箱	本书页码
syms	申明符号变量	符号运算	11
vpa()	直接对符号变量求值	符号运算	11
赋值语句	直接赋值语句	MATLAB	12
函数调用	函数调用语句格式	MATLAB	13
冒号表达式	冒号表达式，可以用于等间距行向量的赋值与子矩阵提取	MATLAB	13
simple()	符号表达式的化简，还可调用 sincos(), numden(), factor(), expand() 等具体化简方法，factor() 函数还可以用于整数的质因数分解	符号运算	17
subs()	符号表达式变量替换	符号运算	17

续表

函数名	函 数 功 能	工具箱	本书页码
latex()	将符号表达式转换成 LaTeX 排版语言支持的字符串	符号运算	18
floor()	该函数可以用于对数值进行取整运算，相应的取整函数还有 round(), fix(), ceil() 等，它们的含义是不同的，具体说明参见表 2-1	MATLAB	表 2-1
rat()	将矩阵的各个数值用最简分式表示	MATLAB	表 2-1
rem()	将矩阵的各个数值取余数	MATLAB	表 2-1
gcd()	求取两个整数的最大公约数，lcm() 求取最小公倍数	符号运算	18
isprime()	判定矩阵内各个整数是否为质数	符号运算	18
for	for 循环结构，和 end 语句共同构成循环，break 语句可以终止本级循环	MATLAB	20
while	while 循环结构	MATLAB	21
if	条件转移语句，可以和 elseif, else 语句连用	MATLAB	22
switch	开关结构，和 case 及 otherwise 语句连用	MATLAB	23
try	试探结构，可以和 catch 语句连用	MATLAB	24
function	函数引导语句	MATLAB	25
inline	inline 函数，可以定义直接可以取值的函数	MATLAB	29
@	匿名函数，功能更强于 inline 函数，为 MATLAB 7.0 提出的新函数	MATLAB	29
plot()	二维直角坐标系曲线绘制	MATLAB	29
set()	MATLAB 对象属性设定函数	MATLAB	32
get()	MATLAB 对象属性提取函数	MATLAB	32
bar()	二维条形图绘制，其他还有 comet(), feather, hist(), polar(), stairs(), compass(), errorbar(), fill(), loglog(), quiver(), stem(), semilogx(), semilogy() 等，具体参见表 2-4	MATLAB	32
ezplot()	二维隐函数曲线绘制函数	MATLAB	33
plot3()	三维曲线绘制函数，其余三维曲线绘制函数包括 stem3(), comet3(), fill3(), bar3() 等	MATLAB	37
meshgrid()	二维或三维网格数据生成	MATLAB	38
mesh()	三维网格曲线绘制	MATLAB	38
surf()	三维表面图形绘制，类似的还有 surfc(), surfl(), waterfall(), contour(), contour3() 等	MATLAB	38
shading	曲面类型设置命令，可以设置成 flat, interp, faceted	MATLAB	38
view()	设置三维图的视角	MATLAB	41

- 本章首先介绍了变量名的命名规则，然后介绍了常用常量、基本语句结构、基本数据结构、矩阵输入、冒号表达式及子矩阵提取等方面的问题，还介绍了 MATLAB 语言的编程基础。
- 介绍了矩阵的各种基本运算方法及 MATLAB 语言实现，包括矩阵的代数运算、逻辑运算和比较运算，还包括基本数论运算和符号运算表达式的定义与化简等内容。

- 介绍了 MATLAB 语言编程的一个很重要的环节，即 MATLAB 语言的基本流程控制语句，如两种循环结构、条件转移结构、开关语句和新型的试探语句等，掌握了这些结构就可以初步编写出基本的 MATLAB 程序。
- MATLAB 语言的最基本程序结构是 MATLAB 的函数结构，本章介绍了函数编写的基本方法及函数编写技巧，如函数的递归调用方法、任意输入输出变量编程方法等，为初步的 MATLAB 语言程序设计打下基础。
- MATLAB 中图形绘制是其一大特色，本章介绍了二维图形和三维图形的绘制方法，可以由已知数据绘制出所需的图形，利用本章介绍的方法就可以直接绘制出各种各样的二维曲线、三维曲线甚至三维曲面与等高线等，还介绍了三维图形的旋转与视角变换等内容。

2.8 习　题

1 启动 MATLAB 环境，并给出语句 `tic, A=rand(500); B=inv(A); norm(A*B-eye(500)), toc`，试运行该语句，观察得出的结果，并利用 `help` 命令对你不熟悉的语句进行帮助信息查询，逐条给出上述程序段与结果的解释。

2 试用符号元素工具箱支持的方式表达多项式 $f(x)=x^5+3x^4+4x^3+2x^2+3x+6$，并令 $x=\dfrac{s-1}{s+1}$，将 $f(x)$ 替换成 s 的函数。

3 用 MATLAB 语句输入矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$

$$\boldsymbol{A}=\begin{bmatrix}1&2&3&4\\4&3&2&1\\2&3&4&1\\3&2&4&1\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}1+4\mathrm{j}&2+3\mathrm{j}&3+2\mathrm{j}&4+1\mathrm{j}\\4+1\mathrm{j}&3+2\mathrm{j}&2+3\mathrm{j}&1+4\mathrm{j}\\2+3\mathrm{j}&3+2\mathrm{j}&4+1\mathrm{j}&1+4\mathrm{j}\\3+2\mathrm{j}&2+3\mathrm{j}&4+1\mathrm{j}&1+4\mathrm{j}\end{bmatrix}$$

前面给出的是 4×4 矩阵，如果给出 $\boldsymbol{A}(5,6)=5$ 命令将得出什么结果？

4 假设已知矩阵 $\boldsymbol{A}$，试给出相应的 MATLAB 命令，将其全部偶数行提取出来，赋给 $\boldsymbol{B}$ 矩阵，用 $\boldsymbol{A}$ =`magic(8)` 命令生成 $\boldsymbol{A}$ 矩阵，用上述的命令检验一下结果是不是正确。

5 用 MATLAB 语言实现下面的分段函数 $y=f(x)=\begin{cases}h, & x>D\\ h/Dx, & |x|\leqslant D\\ -h, & x<-D\end{cases}$ 。

6 用数值方法可以求出 $S=\sum\limits_{i=0}^{63}2^i=1+2+4+8+\cdots+2^{62}+2^{63}$，试不采用循环的形式求出和式的数值解。由于数值方法采用 `double` 形式进行计算的，难以保证有效位数字，所以结果不一定精确。试采用符号运算的方法求该和式的精确值。

7 编写一个矩阵相加函数 `mat_add()`，使其具体的调用格式为 $\boldsymbol{A}$=`mat_add(`$\boldsymbol{A}_1,\boldsymbol{A}_2,\boldsymbol{A}_3,\cdots$`)`，要求该函数能接受任意多个矩阵进行解法运算。

8 自己编写一个 MATLAB 函数，使它能自动生成一个 $m\times m$ 的 Hankel 矩阵，并使其调用格式为 $\boldsymbol{v}$=`[`$h_1,h_2,h_m,h_{m+1},\cdots,h_{2m-1}$`]`; $\boldsymbol{H}$=`myhankel(`$\boldsymbol{v}$`)`。

9 已知 Fibonacci 数列由式 $a_k = a_{k-1} + a_{k-2}$, $k = 3, 4, \cdots$ 可以生成，其中初值为 $a_1 = a_2 = 1$，试编写出生成某项 Fibonacci 数值的 MATLAB 函数，要求：

① 函数格式为 `y=fib(k)`，给出 k 即能求出第 k 项 a_k 并赋给 $\boldsymbol{y}$ 向量；

② 编写适当语句，对输入输出变量进行检验，确保函数能正确调用；

③ 利用递归调用的方式编写此函数。

10 由矩阵理论可知，如果一个矩阵 $\boldsymbol{M}$ 可以写成 $\boldsymbol{M} = \boldsymbol{A} + \boldsymbol{BCB}^{\mathrm{T}}$, 并且其中 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ 为相应阶数的矩阵，则 $\boldsymbol{M}$ 矩阵的逆矩阵可以由下面的算法求出。

$$\boldsymbol{M}^{-1} = \left(\boldsymbol{A} + \boldsymbol{BCB}^{\mathrm{T}}\right)^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\left(\boldsymbol{C}^{-1} + \boldsymbol{B}^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{B}\right)^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{A}^{-1}$$

试根据上面的算法用 MATLAB 语句编写一个函数对矩阵 $\boldsymbol{M}$ 进行求逆，并通过一个小例子来检验该程序，并和直接求逆方法进行精度上的比较。

11 下面给出了一个迭代模型

$$\begin{cases} x_{k+1} = 1 + y_k - 1.4x_k^2 \\ y_{k+1} = 0.3x_k \end{cases}$$

写出求解该模型的 M-函数。如果取迭代初值为 $x_0 = 0$, $y_0 = 0$，那么请进行 30000 次迭代求出一组 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 向量，然后在所有的 x_k 和 y_k 坐标处点亮一个点 (注意不要连线)，最后绘制出所需的图形。**提示** 这样绘制出的图形又称为 Henon 引力线图，它将迭代出来的随机点吸引到一起，最后得出貌似连贯的引力线图。

12 用 MATLAB 语言的基本语句显然可以立即绘制一个正三角形。试结合循环结构，编写一个小程序，在同一个坐标系下绘制出该正三角形绕其中心旋转后得出的一系列三角形，还可以调整旋转步距观察效果。

13 选择合适的步距绘制出图形 $\sin\left(\dfrac{1}{t}\right)$，其中 $t \in (-1, 1)$。

14 对合适的 θ 范围选取分别绘制出下列极坐标图形。

① $\rho = 1.0013\theta^2$， ② $\rho = \cos(7\theta/2)$， ③ $\rho = \sin(\theta)/\theta$， ④ $\rho = 1 - \cos^3(7\theta)$

15 用图解的方式找到下面两个方程构成的联立方程的近似解。

$x^2 + y^2 = 3xy^2$, $x^3 - x^2 = y^2 - y$

16 请分别绘制出 xy 和 $\sin(xy)$ 的三维图和等高线。

17 在图形绘制语句中，若函数值为不定式 `NaN`，则相应的部分不绘制出来。试利用该规律绘制 $z = \sin xy$ 的表面图，并剪切下 $x^2 + y^2 \leqslant 0.5^2$ 的部分。

第3章　微积分问题的计算机求解

Issac Newton 和 Gattfried Wilhelm Leibnitz 创立的微积分学是很多科学分支的基础。单变量与多变量函数微积分、函数极限、级数求和、Taylor 幂级数展开、Fourier 级数展开、常微分方程等问题直接求解是微积分学的重要内容。MATLAB 的符号运算工具箱可以直接求解这样问题的解析解，故在第 3.1 节中侧重介绍基于 MATLAB 语言的微积分问题解析求解方法。第 3.2 节将介绍给定单变量函数与多变量函数的 Taylor 幂级数展开、给定函数的 Fourier 级数逼近方法及一般级数的求和方法等。第 3.5 节中将介绍的两类曲线积分和两类曲面积分及其 MATLAB 求解方法补充了微积分学的计算机求解方式。而第 7 章中将全面介绍的微分方程求解也应该认为是微积分学的一个重要组成部分，同样可以用 MATLAB 语言求解。这部分内容大部分均应该是解析求解和解析推导，属于计算机代数研究的领域，用传统的数值分析方法是不能求解的。对不熟悉计算机代数系统开发的读者来说，用 C 这样的底层语言开发程序进行解析解推导有很大难度，而必须使用计算机数学语言 (如 MATLAB 语言) 完成这类问题的分析与求解。

在实际科学与工程研究中，微积分问题在解析求解时有时也面临困难。例如，若函数本身未知，只通过科学实验测出了一些实验数据，则无法用推导的方式通过数据对其代表的函数求导或求积分，而需要通过数值的方式进行数值微分与数值积分的运算等。在第 3.3 节中将介绍实用的中心差分数值微分算法及其 MATLAB 语言实现，还将介绍多变量函数的偏微分求解方法与实例。在实际应用中还有很多函数积分的解析解不存在，所以需要通过数值积分的算法进行近似。第 3.4 节中将介绍用数值算法介绍函数积分及重积分问题求解方法。若给出的数据点较稀疏，则基于数据直接求取数值微积分会有很大的误差，但可以结合第 8 章中介绍的样条插值方法对其求解。具体内容请参见第 8.2 节。

3.1　微积分问题的解析解

应用 MATLAB 语言的符号运算工具箱，可以很容易求解极限问题、微分问题、积分问题等微积分基本问题的计算机辅助求解方法及应用。利用本节介绍的方法，读者应该能立即具备依赖 MATLAB 语言及其符号运算工具箱中提供的强大函数直接求解一般微积分运算问题的能力。

3.1.1　极限问题的解析解

3.1.1.1　单变量函数的极限

假设已知函数 $f(x)$，则极限问题的一般描述为

$$L = \lim_{x \to x_0} f(x) \tag{3-1-1}$$

其中，x_0 可以是一个确定的值，也可以是无穷大，例如 $x \to \infty$。对某些函数来说，还可以如下定义单边极限 (或称左右极限) 问题。

$$L_1 = \lim_{x \to x_0^-} f(x), \text{ 或 } L_2 = \lim_{x \to x_0^+} f(x) \tag{3-1-2}$$

前者表示 x 从左侧趋近于 x_0 点，所以又称为左极限，后者相应地称为右极限。极限问题在 MATLAB 符号运算工具箱中可以使用 `limit()` 函数直接求出，该函数的调用格式为

L=`limit(fun,` x, x_0`)`	求极限
L=`limit(fun,` x, x_0, `'left'` 或 `'right')`	求单边极限

在求解之前应该先申明自变量 x，再定义极限表达式 `fun`，若 x_0 为 ∞，则可以用 `inf` 直接表示。如果需要求解左右极限问题，还需要给出左右选项。下面将通过例子演示 MATLAB 求解极限的方法。

【例 3-1】 试求解极限问题 $\lim_{x\to\infty} x\left(1+\dfrac{a}{x}\right)^x \sin\dfrac{b}{x}$。

【求解】 利用 MATLAB 语言，应该首先申明 a, b 和 x 为符号变量，然后定义极限式子，最后调用 `limit()` 函数求出给定函数的极限。

```
>> syms x a b; f=x*(1+a/x)^x*sin(b/x); L=limit(f,x,inf)
L =
    exp(a)*b
```

【例 3-2】 试求解单边极限问题 $\lim_{x\to 0^+} \dfrac{e^{x^3}-1}{1-\cos\sqrt{x-\sin x}}$。

【求解】 利用 MATLAB 语言的 `limit()` 函数，可以容易地求取单边极限。

```
>> syms x; limit((exp(x^3)-1)/(1-cos(sqrt(x-sin(x)))),x,0,'right')
ans =
    12
```

用单边极限的方法可求出该函数的极限值为 12。用下面的语句还可以绘制出 $(-0.1, 0.1)$ 区间的函数曲线，如图 3-1 所示。

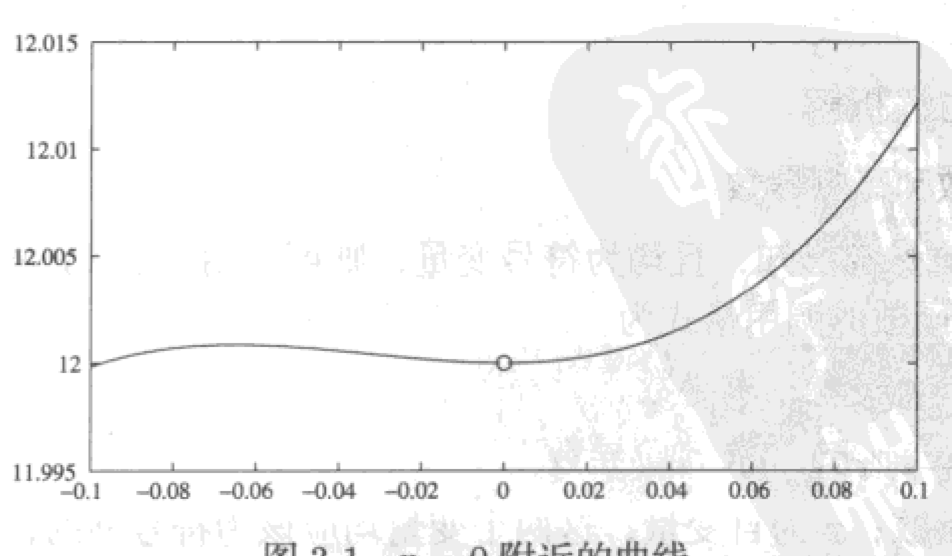


图 3-1 $x = 0$ 附近的曲线

```
>> x=-0.1:0.001:0.1; y=(exp(x.^3)-1)./(1-cos(sqrt(x-sin(x))));
   plot(x,y,'-',[0],[12],'o')
```

可见，对这个例子来说，即使不用单边极限也能求出函数极限值是 12。

```
>> syms x; limit((exp(x^3)-1)/(1-cos(sqrt(x-sin(x)))),x,0)
ans =
    12
```

回顾原始问题，其中采用 $x\to 0^+$ 是因为它可以保证根号内的值为非负数。事实上，即使是负数，$\cos(\mathrm{j}\alpha)=(\mathrm{e}^{\alpha}+\mathrm{e}^{-\alpha})/2$ 也是有定义的，故对这个问题没有影响。但对分段函数来说，单边极限是有意义的。

3.1.1.2 多变量函数的极限

多元函数的极限也可以同样用 MATLAB 中的 `limit()` 函数直接求解。假设有二元函数 $f(x,y)$，若想求出二元函数的极限

$$L=\lim_{\substack{x\to x_0\\ y\to y_0}} f(x,y) \tag{3-1-3}$$

则可以嵌套使用 `limit()` 函数。例如，

L_1=`limit(limit(`f,x,x_0`),`y,y_0`)` 或
L_1=`limit(limit(`f,y,y_0`),`x,x_0`)`

如果 x_0 或 y_0 不是确定的值，而是另一个变量的函数，例如 $x\to g(y)$，则上述的极限求取顺序不能交换。

【例 3-3】 试求出二元函数极限值 $\displaystyle\lim_{\substack{x\to 1/\sqrt{y}\\ y\to\infty}} \mathrm{e}^{-1/(y^2+x^2)}\frac{\sin^2 x}{x^2}\left(1+\frac{1}{y^2}\right)^{x+a^2y^2}$。

【求解】 本例中的问题可以用下面的语句直接解出。

```
>> syms x y a; f=exp(-1/(y^2+x^2))*sin(x)^2/x^2*(1+1/y^2)^(x+a^2*y^2);
   L=limit(limit(f,x,1/sqrt(y)),y,inf)
L =
    exp(a^2)
```

3.1.2 函数导数的解析解

3.1.2.1 函数的导数和高阶导数

如果函数和自变量都已知，且均为符号变量，则可以用 `diff()` 函数解出给定函数的各阶导数。`diff()` 函数的调用格式为

y=`diff(fun,`x`)`　　求导数
y=`diff(fun,`x,n`)`　求 n 阶导数

其中，fun 为给定函数，x 为自变量，这两个变量均应该为符号型的，n 为导数的阶次，若省略 n 则将自动求取一阶导数。

【例 3-4】给定函数 $f(x)=\sin x/(x^2+4x+3)$，试求出 $\mathrm{d}^4 f(x)/\mathrm{d}x^4$。

【求解】这是本书开始时给出的第一个例子。函数的导数和高阶导数问题可以很容易地用 MATLAB 符号运算工具箱语句求解。可以首先申明 x 为符号变量，然后调用 diff() 函数就能直接得出函数的一阶导数。

```
>> syms x; f=sin(x)/(x^2+4*x+3); f1=diff(f); pretty(f1)
                cos(x)       sin(x) (2 x + 4)
             ------------ - ----------------
               2               2          2
             x  + 4 x + 3    (x  + 4 x + 3)
```

从上面得出的结果可以看出，直接采用 MATLAB 显示形式不能得出令人满意的效果。有时需要用 latex() 函数将其转换成专业科学排版语言 LaTeX 的对象，插入 LaTeX 排版语言，得出更好的显示效果为 $\dfrac{\cos x}{x^2+4x+3}-\dfrac{\sin x\,(2x+4)}{(x^2+4x+3)^2}$。

可见，采用 LaTeX 排版效果后得出的结果大大优于 MATLAB 自身的显示，其可读性亦大大提高，故以后的内容中将尽量采用 LaTeX 形式显示得出的结果。

由原函数和得出的一阶导数函数可以对给定自变量向量 $\boldsymbol{x}$ 求出各点处的函数值，并由下面的 MATLAB 命令绘制出来，如图 3-2 所示。从图中可以看出一个函数和其一阶导数之间的关系。

```
>> x1=0:.01:5; y=subs(f,x,x1); y1=subs(f1,x,x1);
   plot(x1,y,x1,y1,':')     % 函数函数及其一阶导数
```

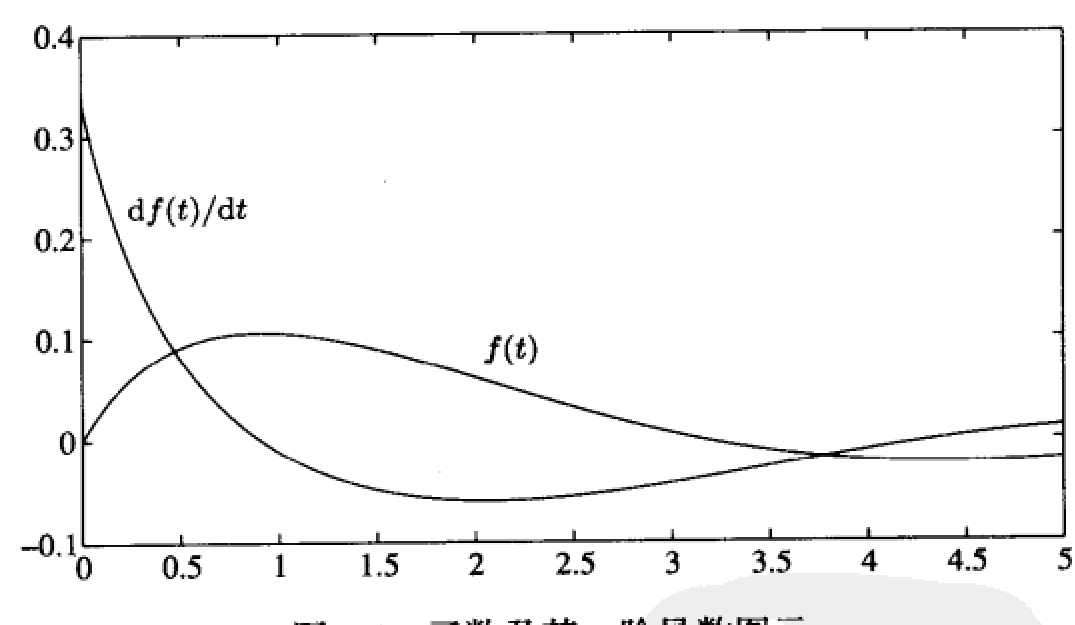


图 3-2 函数及其一阶导数图示

原函数的 4 阶导数可以直接由下面的语句求出。

```
>> f4=diff(f,x,4); latex(f4)    % 求解四阶导数
```

得出的结果比较冗长，所以改用 LaTeX 可以显示出更好的效果如下：

$$\frac{\sin x}{x^2+4x+3}+4\frac{\cos x\,(2x+4)}{(x^2+4x+3)^2}-12\frac{\sin x\,(2x+4)^2}{(x^2+4x+3)^3}+12\frac{\sin x}{(x^2+4x+3)^2}-24\frac{\cos x\,(2x+4)^3}{(x^2+4x+3)^4}$$
$$+48\frac{\cos x\,(2x+4)}{(x^2+4x+3)^3}+24\frac{\sin x\,(2x+4)^4}{(x^2+4x+3)^5}-72\frac{\sin x\,(2x+4)^2}{(x^2+4x+3)^4}+24\frac{\sin x}{(x^2+4x+3)^3}$$

从化简的结果看，单纯采用 simple() 函数得出的化简结果不一定是令人满意的最简结

果，而需要根据具体问题选择合适的化简方法。仔细分析上述的结果，可以发现若按照 $\sin x$ 或 $\cos x$ 单独进行处理，则可能得出最简的结果。这样，分别运行 collect(simple(f4),cos(x)) 和 collect(simple(f4),sin(x))，则可以得出下面给出的更简洁的结果。

$$\frac{\mathrm{d}^4 f(x)}{\mathrm{d}x^4} = 8(x^5+10x^4+26x^3-4x^2-99x-102)\frac{\cos x}{(x^2+4x+3)^4}+$$
$$(x^8+16x^7+72x^6-32x^5-1094x^4-3120x^3-3120x^2+192x+1581)\frac{\sin x}{(x^2+4x+3)^5}$$

MATLAB 现成的 diff() 函数还适合于求解给定函数更高阶的导数。例如，下面给出的命令可以在 5 秒内获得函数的 100 阶导函数。

```
>> tic, diff(f,x,100); toc
Elapsed time is 4.416000 seconds.
```

3.1.2.2　多元函数的偏导数

MATLAB 的符号运算工具箱中并未提供求取偏导数的专门函数，这些偏导数仍然可以通过 diff() 函数直接实现。假设已知二元函数 $f(x,y)$，若想求 $\partial^{m+n} f/(\partial x^m \partial y^n)$，则可以用下面的函数求出。

> *f*=diff(diff(*f*,*x*,*m*),*y*,*n*)，　或
> *f*=diff(diff(*f*,*y*,*n*),*x*,*m*)

【例 3-5】 试求出二元函数 $z=f(x,y)=(x^2-2x)\mathrm{e}^{-x^2-y^2-xy}$ 的偏导数，并用图形表示。

【求解】 用下面的语句可直接求出 $\partial z/\partial x$ 与 $\partial z/\partial y$。

```
>> syms x y
   z=(x^2-2*x)*exp(-x^2-y^2-x*y);
   zx=simple(diff(z,x))
zx =
   -exp(-x^2-y^2-x*y)*(-2*x+2+2*x^3+x^2*y-4*x^2-2*x*y)
>> zy=diff(z,y)
zy =
   -x*(x-2)*(2*y+x)*exp(-x^2-y^2-x*y)
```

其数学形式分别为

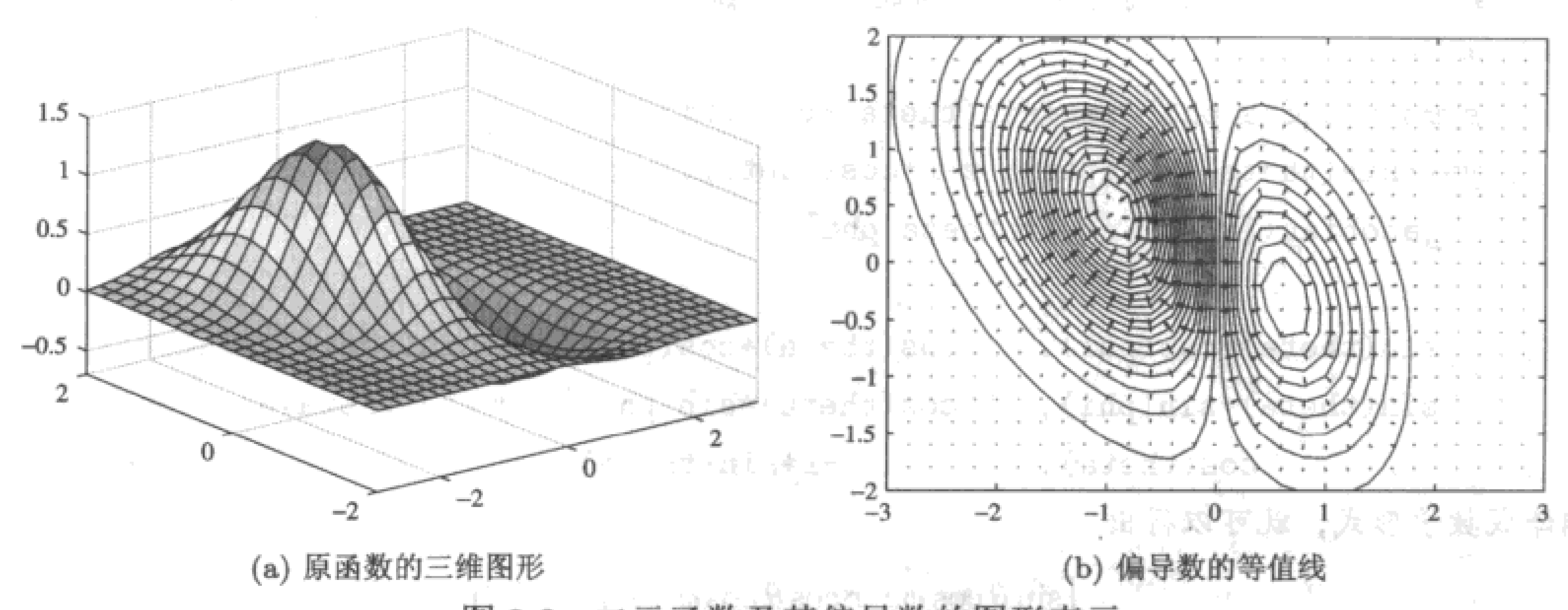
$$\frac{\partial z(x,y)}{\partial x} = -\mathrm{e}^{-x^2-y^2-xy}(-2x+2+2x^3+x^2y-4x^2-2xy)$$
$$\frac{\partial z(x,y)}{\partial y} = -x(x-2)(2y+x)\mathrm{e}^{-x^2-y^2-xy}$$

在 $x\in(-3,3), y\in(-2,2)$ 区域内生成网格，则可以分别得出原函数及其偏导数的数值解。这样，可以直接用下面的语句绘制出原函数的三维曲面，如图 3-3 (a) 所示。

```
>> [x,y]=meshgrid(-3:.2:3,-2:.2:2);
   z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
   surf(x,y,z), axis([-3 3 -2 2 -0.7 1.5]) % 直接绘制三维曲面
```

既然计算出了对两个自变量的一阶偏导数，则可以调用 quiver() 函数绘制出引力线，该引力线可以叠印在以 contour() 函数绘制出的等值线上，如图 3-3 (b) 所示。其中，引力线绘制函数的详细信息可以由 help quiver 命令进一步列出。

```
>> contour(x,y,z,30), hold on    % 绘制等值线
   zx=-exp(-x.^2-y.^2-x.*y).*(-2*x+2+2*x.^3+x.^2.*y-4*x.^2-2*x.*y);
   zy=-x.*(x-2).*(2*y+x).*exp(-x.^2-y.^2-x.*y);      % 偏导的数值解
   quiver(x,y,zx,zy)  % 绘制引力线
```



(a) 原函数的三维图形　　(b) 偏导数的等值线

图 3-3　二元函数及其偏导数的图形表示

【例 3-6】 已知三元函数 $f(x,y,z)=\sin(x^2y)\mathrm{e}^{-x^2y-z^2}$，试求出偏导数 $\partial^4 f(x,y,z)/(\partial x^2\partial y\partial z)$。

【求解】 由下面的语句申明自变量及函数，则可以用 MATLAB 语句立即得出所需的偏导函数。

```
>> syms x y z; f=sin(x^2*y)*exp(-x^2*y-z^2);
   df=diff(diff(diff(f,x,2),y),z); df=simple(df); latex(df);
```

得出的结果很冗长，这里不直接列出，但给出用 LaTeX 语言的表示效果为

$$-4z\mathrm{e}^{-x^2y-z^2}\left[\cos\left(x^2y\right)-10\cos\left(x^2y\right)yx^2+4x^4\sin\left(x^2y\right)y^2+4\cos\left(x^2y\right)x^4y^2-\sin\left(x^2y\right)\right]$$

3.1.2.3 多元函数的 Jacobi 矩阵

假设有 n 个自变量的 m 个函数定义为

$$\left\{\begin{array}{l} y_1=f_1(x_1,x_2,\cdots,x_n) \\ y_2=f_2(x_1,x_2,\cdots,x_n) \\ \quad\vdots \qquad \vdots \\ y_m=f_m(x_1,x_2,\cdots,x_n) \end{array}\right. \tag{3-1-4}$$

则相应的 y_i 对 x_j 求偏导，则得出矩阵

$$\boldsymbol{J}=\begin{bmatrix} \partial y_1/\partial x_1 & \partial y_1/\partial x_2 & \cdots & \partial y_1/\partial x_n \\ \partial y_2/\partial x_1 & \partial y_2/\partial x_2 & \cdots & \partial y_2/\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial y_m/\partial x_1 & \partial y_m/\partial x_2 & \cdots & \partial y_m/\partial x_n \end{bmatrix} \tag{3-1-5}$$

该矩阵又称为 Jacobi 矩阵，它在图像处理、机器人等诸多领域中均是很有用的概念。Jacobi 矩阵可以由 MATLAB 的符号运算工具箱中的 `jacobian()` 函数直接求得。该函数的调用格式为

```
J=jacobian(y, x)
```

其中，$\boldsymbol{x}$ 是自变量构成的向量，$\boldsymbol{y}$ 是由各个函数构成的向量。

【例 3-7】假设有直角坐标和极坐标变换公式为 $x=r\sin\theta\cos\phi$, $y=r\sin\theta\sin\phi$, $z=r\cos\theta$，试推导其 Jacobi 矩阵。

【求解】可以先申明 3 个符号变量并描述 3 个函数，这样可以用下面的语句容易地求解出其 Jacobi 矩阵。

```
>> syms r theta phi; x=r*sin(theta)*cos(phi);
   y=r*sin(theta)*sin(phi); z=r*cos(theta);
   J=jacobian([x y z],[r theta phi])
J =
[   sin(theta)*cos(phi),  r*cos(theta)*cos(phi), -r*sin(theta)*sin(phi)]
[   sin(theta)*sin(phi),  r*cos(theta)*sin(phi),  r*sin(theta)*cos(phi)]
[            cos(theta),          -r*sin(theta),                      0]
```

翻译成数学形式，就可以得出

$$\boldsymbol{J}=\begin{bmatrix}\sin\theta\cos\phi & r\cos\theta\cos\phi & -r\sin\theta\sin\phi\\ \sin\theta\sin\phi & r\cos\theta\sin\phi & r\sin\theta\cos\phi\\ \cos\theta & -r\sin\theta & 0\end{bmatrix}$$

3.1.2.4 隐函数的偏导数

已知隐函数的数学表达式为 $f(x_1,x_2,\cdots,x_n)=0$，则可以通过隐函数对它们的偏导数求出自变量之间的偏导数。具体可以用下面的公式求出 $\partial x_i/\partial x_j$。

$$\frac{\partial x_i}{\partial x_j}=-\frac{\dfrac{\partial}{\partial x_j}f(x_1,x_2,\cdots,x_n)}{\dfrac{\partial}{\partial x_i}f(x_1,x_2,\cdots,x_n)} \tag{3-1-6}$$

由于 f 对 x_i, x_j 的偏导数可以分别由 `diff()` 函数求出，故整个偏导数可以由它们的除法获得，所以这样的问题可以用 MATLAB 语句容易地得出。

```
F=-diff(f,xj)/diff(f,xi)
```

【例 3-8】考虑例 3-5 中给出的二元函数 $z=f(x,y)=(x^2-2x)\mathrm{e}^{-x^2-y^2-xy}$，试求 $\partial y/\partial x$。

【求解】根据式 (3-1-6) 可以立即得出所需偏导数 $\partial y/\partial x$ 为

```
>> syms x y; f=(x^2-2*x)*exp(-x^2-y^2-x*y);
   pretty(-simple(diff(f,x)/diff(f,y)))
```

用 LaTeX 表示结果为

$$-\frac{-2x+2+2x^3+x^2y-4x^2-2xy}{x\,(x-2)\,(2y+x)}$$

3.1.2.5 参数方程的导数

若已知参数方程 $y=f(t), x=g(t)$，则 $\dfrac{\mathrm{d}^k y}{\mathrm{d}x^k}$ 可以由下面的 MATLAB 语句直接求出。

```
diff(f,t,k)/diff(g,t,k)
```

【例 3-9】已知参数方程 $y=\dfrac{\sin t}{(t+1)^3}, x=\dfrac{\cos t}{(t+1)^3}$，试求 $\dfrac{\mathrm{d}^4 y}{\mathrm{d}x^4}$。

【求解】由前面给出的函数调用格式，可以立即得出所需的高阶导数。

```
>> syms t; y=sin(t)/(t+1)^3; x=cos(t)/(t+1)^3;
   latex(diff(y,t,4)/diff(x,t,4))
```

由 LaTeX 排版格式可以自动得出如下的结果。

$$\frac{\mathrm{d}^4 y}{\mathrm{d}x^4}=\frac{\dfrac{\sin t}{(t+1)^3}+12\dfrac{\cos t}{(t+1)^4}-72\dfrac{\sin t}{(t+1)^5}-240\dfrac{\cos t}{(t+1)^6}+360\dfrac{\sin t}{(t+1)^7}}{\dfrac{\cos t}{(t+1)^3}-12\dfrac{\sin t}{(t+1)^4}-72\dfrac{\cos t}{(t+1)^5}+240\dfrac{\sin t}{(t+1)^6}+360\dfrac{\cos t}{(t+1)^7}}$$

3.1.3 积分问题的解析解

3.1.3.1 不定积分的推导

MATLAB 符号运算工具箱中提供了一个 int() 函数，可以直接用来求取符号函数的不定积分。该函数的调用格式为

```
F=int(fun,x)
```

如果被积函数 fun 中只有一个变量，则调用语句中的 x 可以省略。另外，该函数得出的结果 $F(x)$ 是积分原函数，实际的不定积分应该是 $F(x)+C$ 构成的函数族。其中，C 是任意常数。

对于可积的函数，MATLAB 符号运算工具箱提供的 int() 函数可以用计算机代替繁重的手工推导，立即得出原始问题的解。而对于不可积的函数来说，当然 MATLAB 也是无能为力的。下面将通过例子介绍该函数使用方法及应用。

【例 3-10】考虑例 3-4 中给出的问题，用 diff() 函数可以直接求 $f(x)$ 函数的一阶导数。现在对得出的导数再进行积分，试检验是否可以得出一致的结果。

【求解】先定义原函数并对其求导，然后再对导数进行积分，则

```
>> syms x; y=sin(x)/(x^2+4*x+3); y1=diff(y);   % 对函数求导
   y0=int(y1); latex(y0)     % 对导数积分应该得出原函数
```

得出的 LaTeX 表示为

$$1/2\frac{\sin(x)}{x+1}-1/2\frac{\sin(x)}{x+3}$$

现在对原函数求 4 阶导数，再对结果进行 4 次积分，则可以用下面语句判定正确性。

```
>> y4=diff(y,4); y0=int(int(int(int(y4))));
   latex(simple(y0))
```

仍用 LaTeX 表示为

$$\frac{\sin(x)}{(x+1)(x+3)}$$

【例 3-11】 试证明

$$\int x^3\cos^2(ax)\mathrm{d}x=\frac{x^4}{8}+\left(\frac{x^3}{4a}-\frac{3x}{8a^3}\right)\sin(2ax)+\left(\frac{3x^2}{8a^2}-\frac{3}{16a^4}\right)\cos(2ax)+C$$

【求解】 用 MATLAB 语言的符号运算工具箱可以直接得出下面的化简结果。

```
>> syms a x; f=simple(int(x^3*cos(a*x)^2,x))
f =
  1/16*(4*a^3*x^3*sin(2*a*x)+2*a^4*x^4+6*a^2*x^2*cos(2*a*x)-
         6*a*x*sin(2*a*x)-3*cos(2*a*x)-3)/a^4
```

然而，从得出的结果很难看出它是否和等式右侧完全一致，这样需要将等式右侧的表达式也输入到 MATLAB 工作空间，将二者相减并进行化简，从而得出如下结果。

```
>> f1=x^4/8+(x^3/(4*a)-3*x/(8*a^3))*sin(2*a*x)+...
       (3*x^2/(8*a^2)-3/(16*a^4))*cos(2*a*x);
   simple(f-f1)  % 求两个结果的差
ans =
  -3/16/a^4
```

可见，二者并非完全相等，幸好得出的差为一个常数项，即使两种方法得出的积分原函数有差距，但因为形成积分函数族时需要加一个任意常数 C，故可以认为题中的等式得证。

【例 3-12】 考虑两个不可积问题 $f(x)=\mathrm{e}^{x^2}$ 与 $g(x)=x\sin(ax^4)\mathrm{e}^{x^2/2}$ 的积分问题求解。

【求解】 首先考虑 $f(x)=\mathrm{e}^{x^2}$ 的不定积分求解。用 MATLAB 语言可以给出下面的语句。

```
>> syms x; int(exp(-x^2/2))
ans =
   1/2*pi^(1/2)*2^(1/2)*erf(1/2*2^(1/2)*x)
```

该积分虽然不可积，但可以用数学方法定义一个符号 $\mathrm{erf}(x)=\dfrac{2}{\sqrt{\pi}}\displaystyle\int_0^x \mathrm{e}^{-t^2}\mathrm{d}t$，这样似乎可以写出积分的解析表达式。

再考虑一个纯粹不可积的函数 $g(x)=x\sin(ax^4)\mathrm{e}^{x^2/2}$，用 MATLAB 语句可以尝试对其直接求积分，得出如下不可积的信息。

```
>> syms a x; int(x*sin(a*x^4)*exp(x^2/2))
Warning: Explicit integral could not be found.
> In sym.int at 58
ans =
   int(x*sin(a*x^4)*exp(1/2*x^2),x)
```

3.1.3.2 定积分与无穷积分计算

有些函数不定积分可能不存在，但在实际应用中需要求取它的具体定积分值或无穷积分的值。例如，前面定义的 $\mathrm{erf}(x)$ 函数虽然不定积分不可直接求解，但需要得出 $\mathrm{erf}(1.5)$ 的值，则需要进行定积分运算。在 MATLAB 语言中仍然可以使用 `int()` 函数来求解定积分或无穷积分问题，该函数的具体的调用格式为

```
I=int(f, x, a, b)
```

其中，x 为自变量，(a,b) 为定积分的积分区间，求解无穷积分时，允许将 a, b 设置成 -Inf 或 Inf，如果得出的结果不是确切的数值，还可以用 vpa() 函数得出定积分的解。

【例 3-13】仍考虑例 3-12 中的积分式子，试求出当 $a=0$，$b=1.5$ 或 ∞ 时的定积分值。

【求解】若要求解该问题，需要给出如下的 MATLAB 语句。

```
>> syms x; I1=int(exp(-x^2/2),x,0,1.5)
I1 =
     1/2*erf(3/4*2^(1/2))*2^(1/2)*pi^(1/2)
>> vpa(I1,70)
ans =
    1.085853317666016569702419076542265042534236293532156326729917229308528
>> I2=int(exp(-x^2/2),x,0,inf)
ans =
   1/2*2^(1/2)*pi^(1/2)
```

【例 3-14】试求解函数边界的定积分问题 $I(t)=\displaystyle\int_{\cos t}^{e^{2t}}\frac{-2x^2+1}{(2x^2-3x+1)^2}\mathrm{d}x$。

【求解】MATLAB 提供的 int() 函数还可以求解函数积分区域的定积分问题，题中的定积分可以由下面的 MATLAB 语句直接求解。

```
>> syms x t; f=(-2*x^2+1)/(2*x^2-3*x+1)^2;
   I=simple(int(f,x,cos(t),exp(-2*t))), latex(I)
```

由于直接显式的结果很冗长，所以用自动转换出的 LaTeX 表示为

$$I(t)=-\frac{(2\mathrm{e}^{-2t}\cos t-1)(\mathrm{e}^{-2t}-\cos t)}{(\mathrm{e}^{-2t}-1)(2\mathrm{e}^{-2t}-1)(\cos t-1)(2\cos t-1)}$$

3.1.3.3 多重积分问题的 MATLAB 求解

多重积分问题也可以在 MATLAB 语言环境中直接求解，但需要根据实际情况先选择积分顺序，可积的部分作为内积分，然后再处理外积分。每步积分均采用 int() 函数处理，如果交换积分顺序后仍然不能积出解析解，则说明原积分问题没有解析解，而需要采用数值方法求解原始的积分问题。多重积分的数值解法将在第 3.4.3 节中介绍。

【例 3-15】考虑下面的三元函数 $F(x,y,z)$

$$-4z\mathrm{e}^{-x^2y-z^2}\left[\cos\left(x^2y\right)-10\cos\left(x^2y\right)yx^2+4x^4\sin\left(x^2y\right)y^2+4\cos\left(x^2y\right)x^4y^2-\sin\left(x^2y\right)\right]$$

试求出 $\displaystyle\iiint F(x,y,z)\mathrm{d}x^2\mathrm{d}y\mathrm{d}z$。

【求解】事实上，此函数是例 3-6 中给出 $f(x,y,z)$ 经偏导运算得出，故需要对求导过程进行逆向运算，还原回原函数的结果。

对该函数进行积分。先对 z 积分一次，对 y 积分一次，再连续对 x 积分两次，经过化简，则得出如下结果。

```
>> syms x y z; f0=-4*z*exp(-x^2*y-z^2)*(cos(x^2*y)-10*cos(x^2*y)*y*x^2+...
```

```
    4*sin(x^2*y)*x^4*y^2+4*cos(x^2*y)*x^4*y^2-sin(x^2*y));
    f1=int(f0,z); f1=int(f1,y); f1=int(f1,x); f1=simple(int(f1,x))
f1 =
    sin(x^2*y)*exp(-x^2*y-z^2)
```

可见，经化简按照上述给出的积分顺序可以得出原函数，和例 3-6 中给出的原函数完全一致。现在考虑改变积分求解顺序，变成 $z \to x \to x \to y$，则可以得出如下结果。

```
>> f2=int(f0,z); f2=int(f2,x); f2=int(f2,x); f2=simple(int(f2,y))
f2 =
    2*exp(-x^2*y-z^2)*tan(1/2*x^2*y)/(1+tan(1/2*x^2*y)^2)
```

可见，得出的化简结果显然不同于原函数。将其和原函数理论值相减，则可以得出误差为 0，表明二者实际上是完全一致的，但由于积分顺序选择不同，得不出实际的最简形式。

```
>> simple(f1-f2)
ans =
      0
```

【例 3-16】 试求解三重定积分问题 $\int_0^1\int_0^{\pi}\int_0^{\pi} 4xze^{-x^2y-z^2}\mathrm{d}z\mathrm{d}y\mathrm{d}x$。

【求解】 用如下的定积分求解语句可以立即得出三重积分的结果为

```
>> syms x y z
   int(int(int(4*x*z*exp(-x^2*y-z^2),x,0,2),y,0,pi),z,0,pi)
ans =
     pi*Ei(1,4*pi)*(1/pi-1/pi*exp(-pi^2))+pi*log(pi)*(1/pi-1/pi*exp(-pi^2))+
     pi*eulergamma*(1/pi-1/pi*exp(-pi^2))+2*pi*log(2)*(1/pi-1/pi*exp(-pi^2))
```

其中，eulergamma 为 Euler 常数 γ，$\mathrm{Ei}(n,z)$ 为指数积分，即 $\mathrm{Ei}(n,z)=\int_1^{\infty}\mathrm{e}^{-zt}t^{-n}\mathrm{d}t$。该函数虽然解析不可积，但可以求出其数值解。这样，原始问题的精确数值解可以如下得出。

```
>> vpa(ans,60)
ans =
    3.10807940208541272283461464767138521019142306317021863483586
```

3.2 函数的级数展开与级数求和问题求解

本节将介绍给定的单变量函数与多变量函数的 Taylor 幂级数展开、各种函数的 Fourier 级数展开、有穷级数与无穷级数求和等问题的计算机求解方法。

3.2.1 Taylor 幂级数展开

3.2.1.1 单变量函数的 Taylor 幂级数展开

如果在 $x=0$ 点附近进行 Taylor 幂级数展开，则得出

$$f(x)=a_1+a_2x+a_3x^2+\cdots+a_kx^{k-1}+o(x^k) \tag{3-2-1}$$

其中，系数 a_i 可以由下面的公式求出。

$$a_i=\frac{1}{i!}\lim_{x\to 0}\frac{\mathrm{d}^{i-1}}{\mathrm{d}x^{i-1}}f(x),\ i=1,2,3,\cdots \tag{3-2-2}$$

该幂级数展开又称为 Maclaurin 展开，若关于 $x=a$ 点进行展开，则可以得出

$$f(x)=b_1+b_2(x-a)+b_3(x-a)^2+\cdots+b_k(x-a)^{k-1}+o[(x-a)^k] \tag{3-2-3}$$

其中，各个系数 b_i 可以如下求出。

$$b_i=\frac{1}{i!}\lim_{x\to a}\frac{\mathrm{d}^{i-1}}{\mathrm{d}x^{i-1}}f(x),\ i=1,2,3,\cdots \tag{3-2-4}$$

Taylor 幂级数展开可以用符号运算工具箱的 `taylor()` 函数直接导出。该函数的调用格式为

```
taylor(f, x, k)      % 按 x=0 进行 Taylor 幂级数展开
taylor(f, x, k, a)   % 按 x=a 进行 Taylor 幂级数展开
```

其中，f 为函数的符号表达式，x 为自变量，若函数只有一个自变量，则 x 可以省略。k 为需要展开的项数，默认值为 6 项。还可以给出 a 参数，表明需要获得关于 $x=a$ 的幂级数展开。下面将通过例子演示 Taylor 幂级数展开的方法。

【例 3-17】 仍考虑例 3-4 中给出的函数 $f(x)=\sin x/(x^2+4x+3)$，试求出该函数的 Taylor 幂级数展开的前 9 项，并关于 $x=2$ 和 $x=a$ 分别进行原函数的 Taylor 幂级数展开。

【求解】 先用下面的语句输入已知的函数，这样就可以调用 `taylor()` 函数导出其 Taylor 幂级数展开的前 9 项。

```
>> syms x; f=sin(x)/(x^2+4*x+3);
   y1=taylor(f,x,9); latex(y1)
```

这样，用 LaTeX 可以显示如下结果。

$$f(x)=\frac{1}{3}x-\frac{4}{9}x^2+\frac{23}{54}x^3-\frac{34}{81}x^4+\frac{4087}{9720}x^5-\frac{3067}{7290}x^6+\frac{515273}{1224720}x^7-\frac{386459}{918540}x^8+\cdots$$

现在计算关于 $x=2$ 的 Taylor 幂级数展开的前若干项仍然能用 MATLAB 导出，可以使用如下语句。

```
>> taylor(y,x,9,2)  % 结果冗长，不列出
```

用 LaTeX 可以列出其中的前 4 项为

$$\frac{\sin 2}{15}+\left(\frac{\cos 2}{15}-\frac{8\sin 2}{225}\right)(x-2)-\left(\frac{127\sin 2}{6750}+\frac{8\cos 2}{225}\right)(x-2)^2+\left(\frac{23\cos 2}{6750}+\frac{628\sin 2}{50625}\right)(x-2)^3$$

若想导出关于某一点 $x=a$ 的 Taylor 幂级数展开，则可以给出如下语句。

```
>> syms a; taylor(y,x,5,a)  % 结果较冗长，显示从略
```

考虑到篇幅，这里只显示展开表达式的前 3 项为

$$\frac{\sin a}{a^2+3+4a}+\left[\frac{\cos a}{a^2+3+4a}-\frac{(4+2a)\sin a}{(a^2+3+4a)^2}\right](x-a)+\left[-\frac{\sin a}{(a^2+3+4a)^2}-\frac{\sin a}{2(a^2+3+4a)}\right.$$

$$-\frac{(a^2\cos a+3\cos a+4a\cos a-4\sin a-2a\sin a)(4+2a)}{(a^2+3+4a)^3}\Bigg](x-a)^2$$

【例 3-18】试对正弦函数 $y=\sin x$ 进行 Taylor 幂级数展开，观察不同阶次下的近似效果。

【求解】根据要求，可以给出如下的 MATLAB 语句，则可以用循环的形式得出各次 Taylor 幂级数展开，得出如图 3-4 所示的拟合曲线。若拟合的阶次较低，则拟合效果较好的区间较小。增大拟合阶次，则拟合较好的区域将明显增大。对本例来说，若选择 $n=16$，则在 $(-2\pi,2\pi)$ 区间内的拟合效果将很理想。

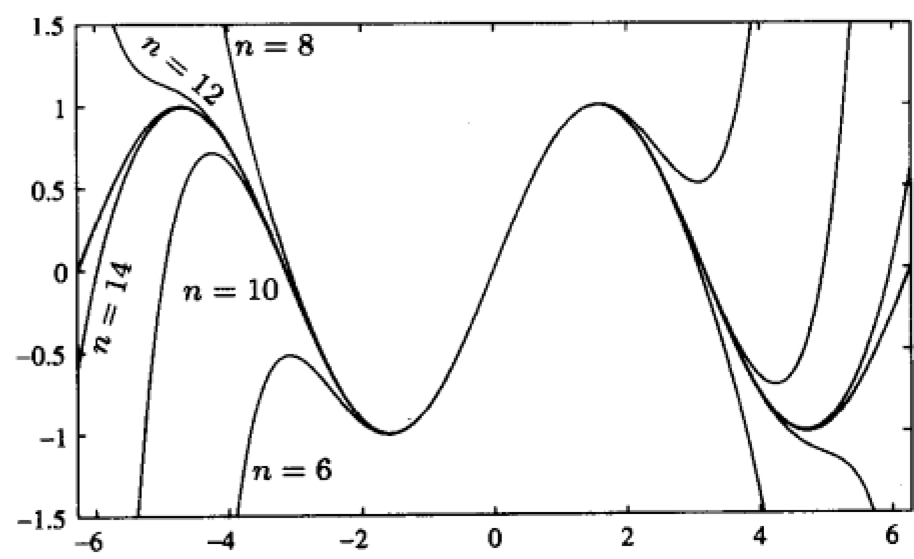


图 3-4 正弦函数的 Taylor 幂级数近似比较

```
>> x0=-2*pi:0.01:2*pi; y0=sin(x0); syms x; y=sin(x);
   plot(x0,y0), axis([-2*pi,2*pi,-1.5,1.5]); hold on
   for n=[8:2:16]
      p=taylor(y,x,n), y1=subs(p,x,x0); line(x0,y1)
   end
```

用 `latex(p)` 则将得出 16 阶 Taylor 幂级数展开式为

$$\sin x\simeq x-\frac{1}{6}x^3+\frac{1}{120}x^5-\frac{1}{5040}x^7+\frac{1}{362880}x^9-\frac{1}{39916800}x^{11}+\frac{1}{6227020800}x^{13}-\frac{1}{1307674368000}x^{15}$$

3.2.1.2 多变量函数的 Taylor 幂级数展开

多变量函数 $f(x_1,x_2,\cdots,x_n)$ 的 Taylor 幂级数展开可以写成

$$\begin{aligned}f(x_1,\cdots,x_n)=f(a_1,\cdots,a_n)+&\\ &\left[(x_1-a_1)\frac{\partial}{\partial x_1}+\cdots+(x_n-a_n)\frac{\partial}{\partial x_n}\right]f(a_1,\cdots,a_n)+\\ &\frac{1}{2!}\left[(x_1-a_1)\frac{\partial}{\partial x_1}+\cdots+(x_n-a_n)\frac{\partial}{\partial x_n}\right]^2 f(a_1,\cdots,a_n)+\cdots+\\ &\frac{1}{k!}\left[(x_1-a_1)\frac{\partial}{\partial x_1}+\cdots+(x_n-a_n)\frac{\partial}{\partial x_n}\right]^k f(a_1,\cdots,a_n)+\cdots\end{aligned}\tag{3-2-5}$$

其中，$a_1,\cdots,a_n$ 为 Taylor 幂级数展开的中心点。为避免歧意，这里的式子应该理解为先对 f 函数求导，再取 $a_1,a_2,\cdots,a_n$ 点的导函数值。MATLAB 的符号运算工具箱并未提供

计算多变量函数 Taylor 幂级数展开的直接函数，但可以用 MATLAB 基本的语句计算该展开。另外，可以调用 Maple 语言中的 mtaylor() 函数来直接求取多变量函数的 Taylor 幂级数展开。该函数的调用格式为

```
F=maple('mtaylor',f,'[x1,···,xn]',k)            根据原点展开
F=maple('mtaylor',f,'[x1=a1,···,xn=an]',k)      根据 (a1,···,an) 点展开
```

其中，$k-1$ 为展开的最高阶次，f 为原多变量函数。注意，该函数调用时自变量的引号不能省略，该调用格式将原封不动地将这些信息传递给 Maple 语言。

【例 3-19】 试求例 3-5 中给出函数 $z=f(x,y)=(x^2-2x)\mathrm{e}^{-x^2-y^2-xy}$ 的各种 Taylor 幂级数展开。

【求解】 使用给出的函数就可以立即得出关于原点的 Taylor 幂级数展开。

```
>> syms x y; f=(x^2-2*x)*exp(-x^2-y^2-x*y);
   F=maple('mtaylor',f,'[x,y]',8)
   latex(collect(F,x))      % 求出关于 x 的多项式
```

其数学表示形式如下：

$$f(x,y)=\frac{x^7}{3}+\left(y+\frac{1}{2}\right)x^6+\left(2y^2+y-1\right)x^5+\left(\frac{7y^3}{3}-1-2y+\frac{3y^2}{2}\right)x^4+$$
$$\left(2y^4+2-y-3y^2+y^3\right)x^3+\left(1-2y^3-y^2+2y+y^5+\frac{y^4}{2}\right)x^2+\left(-2+\frac{y^6}{3}+2y^2-y^4\right)x$$

现在求取关于 $x=1,y=a$ 的幂级数展开，则需要给出语句

```
>> syms a; F=maple('mtaylor',f,'[x=1,y=a]',3);
```

便可以得出如下的幂级数展开式。

$$f(x,y)=-\mathrm{e}^{-1-a-a^2}-\mathrm{e}^{-1-a-a^2}(-2-a)(x-1)-\mathrm{e}^{-1-a-a^2}(-2a-1)(y-a)+$$
$$\left[-\mathrm{e}^{-1-a-a^2}\left(1+2a+\frac{a^2}{2}\right)+\mathrm{e}^{-1-a-a^2}\right](x-1)^2-$$
$$\mathrm{e}^{-1-a-a^2}\left(5a+1+2a^2\right)(y-a)(x-1)-\mathrm{e}^{-1-a-a^2}\left(-\frac{1}{2}+2a+2a^2\right)(y-a)^2+\cdots$$

其实，Maple 的 mtaylor() 函数也可以求出单变量 x 或 y 的 Taylor 幂级数展开。该函数调用起来一样容易。

```
>> F=maple('mtaylor',f,'[x=a]',3);
```

得出的结果用 LaTeX 表示为

$$f(x,y)=(a^2-2a)\mathrm{e}^{-a^2-y^2-ay}+\left[(a^2-2a)(-2a-y)+(2a-2)\right]\mathrm{e}^{-a^2-y^2-ay}(x-a)+$$
$$\left[(a^2-2a)(-1+2a^2+2ay+y^2/2)+1+(2a-2)(-2a-y)\right]\mathrm{e}^{-a^2-y^2-ay}(x-a)^2+\cdots$$

其实，从上面两个结果看，展开式的系数应该还能进一步化简。

3.2.2 Fourier 级数展开

给定周期性数学函数 $f(x)$，其中，$x\in[-L,L]$，且周期为 $T=2L$，可以人为地对该函数在其他区间上进行周期延拓，使得 $f(x)=f(kT+x)$，k 为任意整数，这样可以根据

需要将其写成下面的级数形式。

$$f(x)=\frac{a_0}{2}+\sum_{n=1}^{\infty}\left(a_n\cos\frac{n\pi}{L}x+b_n\sin\frac{n\pi}{L}x\right) \tag{3-2-6}$$

其中，

$$\begin{cases} a_n=\dfrac{1}{L}\displaystyle\int_{-L}^{L}f(x)\cos\frac{n\pi x}{L}\mathrm{d}x, & n=0,1,2,\cdots \\ b_n=\dfrac{1}{L}\displaystyle\int_{-L}^{L}f(x)\sin\frac{n\pi x}{L}\mathrm{d}x, & n=1,2,3,\cdots \end{cases} \tag{3-2-7}$$

该级数称为 Fourier 级数，而 a_n,b_n 又称为 Fourier 系数。若 $x\in(a,b)$，则可以计算出 $L=(b-a)/2$，引入新变量 $\hat{x}$，使得 $x=\hat{x}+L+a$，则可以将 $f(\hat{x})$ 映射成 $(-L,L)$ 区间上的函数，可以对之进行 Fourier 级数展开，再将 $\hat{x}=x-L-a$ 转换成 x 的函数即可。

MATLAB 和 Maple 语言均未直接提供求解 Fourier 系数与级数的现成函数。其实由上述公式不难编写出解析或数值的 Fourier 级数求解函数。其中解析函数如下：

```
function [A,B,F]=fseries(f,x,n,a,b)
if nargin==3, a=-pi; b=pi; end
L=(b-a)/2; if a+b, f=subs(f,x,x+L+a); end
A=int(f,x,-L,L)/L; B=[]; F=A/2;
for i=1:n
    an=int(f*cos(i*pi*x/L),x,-L,L)/L;
    bn=int(f*sin(i*pi*x/L),x,-L,L)/L; A=[A, an]; B=[B,bn];
    F=F+an*cos(i*pi*x/L)+bn*sin(i*pi*x/L);
end
if a+b, F=subs(F,x,x-L-a); end
```

该函数的调用格式为

[*A*,*B*,*F*]=fseries(*f*,*x*,*n*,*a*,*b*)

其中，f 为给定函数，x 为自变量，n 为展开项数，a,b 为 x 的区间，可以省略取其默认值 $[-\pi,\pi]$，得出的 $\boldsymbol{A}$, $\boldsymbol{B}$ 为 Fourier 系数，F 为展开式。仿照解析解 `fseries()` 函数其实不难写出其数值版，有兴趣的读者可以自己试一试。

【例 3-20】 试求给定函数 $y=x(x-\pi)(x-2\pi), x\in(0,2\pi)$ 的 Fourier 级数展开。

【求解】 上述给定函数的 Fourier 级数展开可以很自然地用下面的语句得出。

```
>> syms x; f=x*(x-pi)*(x-2*pi);
   [A,B,F]=fseries(f,x,12,0,2*pi); latex(F)
```

这样，可以得出前 12 项的 Fourier 级数展开为

$$f(x)=12\sin x+\frac{3}{2}\sin 2x+\frac{4}{9}\sin 3x+\frac{3}{16}\sin 4x+\frac{12}{125}\sin 5x+\frac{1}{18}\sin 6x+\frac{12}{343}\sin 7x+ \\ \frac{3}{128}\sin 8x+\frac{4}{243}\sin 9x+\frac{3}{250}\sin 10x+\frac{12}{1331}\sin 11x+\frac{1}{144}\sin 12x$$

其实，该展开的解析表达式为 $f(x)=\sum_{n=1}^{\infty}\dfrac{12}{n^3}\sin nx$。

【例 3-21】考虑 $(-\pi,\pi)$ 区间的方波信号，假设 $x\geqslant 0$ 时 $y=1$，否则 $y=-1$，试对该方波信号进行 Fourier 级数拟合，并观察用多少项能有较好的拟合效果。

【求解】给定的函数可以由 $f(x)=|x|/x$ 表示，故由下面语句可以容易地生成 x-轴数据点，求出理论的方波数值，并通过不同阶次的 Fourier 级数展开去拟合原来的方波函数。得出的曲线如图 3-5 所示。

```
>> syms x; f=abs(x)/x;  % 定义方波信号
   xx=[-pi:pi/200:pi]; xx=xx(xx~=0); xx=sort([xx,-eps,eps]); % 剔除零点
   yy=subs(f,x,xx); plot(xx,yy), hold on  % 绘制出理论值并保持坐标系
   for n=1:20
      [a,b,f1]=fseries(f,x,n); y1=subs(f1,x,xx); plot(xx,y1)
   end
```

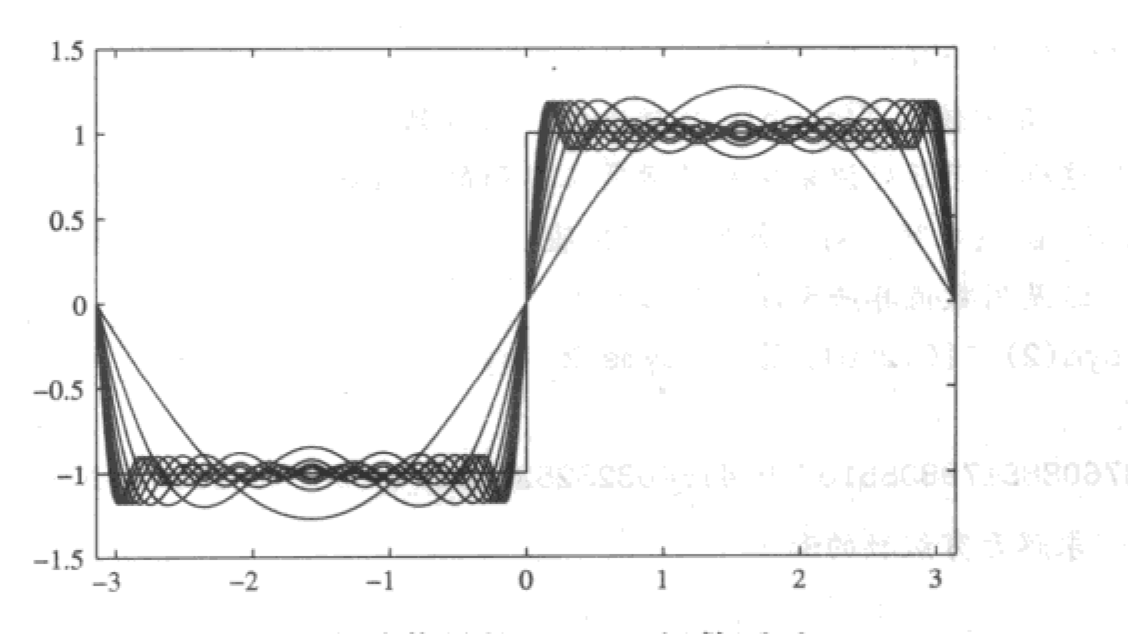


图 3-5 方波信号的 Fourier 级数逼近

从得出的结果看，当阶次等于 10 左右就能得出较好的拟合，再增加阶次也不会有显著的改善效果。取 $n=14$，则 Taylor 级数展开可以由下面的语句具体得出。

```
>> [a,b,f1]=fseries(f,x,14); latex(f1)
```

用 LaTeX 排版语言可以具体表示成

$$f(x)\simeq 4\frac{\sin x}{\pi}+\frac{4\sin 3x}{3\pi}+\frac{4\sin 5x}{5\pi}+\frac{4\sin 7x}{7\pi}+\frac{4\sin 9x}{9\pi}+\frac{4\sin 11x}{11\pi}+\frac{4\sin 13x}{13\pi}$$

从该结果可以总结出一般的展开公式为 $f(x)=\dfrac{4}{\pi}\sum_{k=1}^{\infty}\dfrac{\sin(2k-1)x}{2k-1}$。

3.2.3 级数求和的计算

符号运算工具箱中提供的 symsum() 可以用于已知通项的有穷或无穷级数的和。该函数的调用格式为

S=symsum(f_k,k,k_0,k_n)

其中，f_k 为级数的通项，k 为级数自变量，k_0 和 k_n 为级数求和的起始项与终止项，并可

以将起始或终止项设置成无穷量 inf。该函数可以得出

$$S=\sum_{k=k_0}^{k_n} f_k \tag{3-2-8}$$

如果给出的 f_k 变量中只含有一个变量，则在函数调用时可以省略 k 量。

【例 3-22】计算有限项级数求和

$$S=2^0+2^1+2^2+2^3+2^4+\cdots+2^{62}+2^{63}=\sum_{i=0}^{63}2^i$$

【求解】用数值计算方法可以由下面语句得出结果。

```
>> format long; sum(2.^[0:63])
ans =
   1.844674407370955e+019
```

由于数值计算中使用了 double 数据类型，至多只能保留 16 位有效数字，所以得出的结果不是很精确。对这样的问题应该采用符号运算工具箱的 symsum() 函数，或至少将 2 定义为符号量，就可以用 sum() 函数求解。对原始问题稍扩展一步，一直到第 201 项的级数求和可以用下面的语句精确求出，这是用数值算法无法精确做到的。

```
>> sum(sym(2).^[0:200]) % 或 syms k; symsum(2^k,0,200)
ans =
  3213876088517980551083924184682325205044405987565585670602751
```

【例 3-23】试求解无穷级数的和

$$S=\frac{1}{1\times 4}+\frac{1}{4\times 7}+\frac{1}{7\times 10}+\cdots+\frac{1}{(3n-2)(3n+1)}+\cdots$$

【求解】如果想借助 MATLAB 的符号运算工具箱，则可以立即得如下结果。

```
>> syms n; s=symsum(1/((3*n-2)*(3*n+1)),n,1,inf)
s =
   1/3
```

此级数求和亦可以用数值方法求得。假设求前 10000000 项的和，这时可以求出级数的和。但可以看出，得出无穷级数的和与解析解间存在很大差异，这个差异就是 double 数据类型引起，它不能保留任意多小数位。

```
>> m=1:10000000; s1=sum(1./((3*m-2).*(3*m+1)));
   format long; s1 % 以长型方式显示得出的结果
s1 =
   0.33333332222165
```

可见，即使选择的累加项数极多，耗时很长，这样得出的结果仍然有较大误差，达到 10^{-6} 级。从通项上看，当 $m=10^7$ 时，通项值 10^{-15} 级，从表面上可能得出结论，似乎累加的结果误差不会太大。其实不然，由于双精度数值的有效数位数有限，只有 16 位，所以计算通项时 16 位

后的数字加到累加量上就消失了，这就是数值分析中经常所说的“大数吃小数”的现象，所以采用纯数值的方法，即使取再多的位数也不能精确得出正确的结果。

【例 3-24】试求解含有变量的无穷级数的和

$$J=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)(2x+1)^{2n+1}}$$

【求解】前面介绍的例子都是数值的例子，直接采用累加的方式就可以近似求出结果。这里给出的求和问题是一个含有变量 x 的函数级数，所以仅靠数值运算的方式是不可能得出该级数的和，而必须采用符号运算工具箱求解该问题，这需要给出下面的命令。

```
>> syms n x
   s1=symsum(2/((2*n+1)*(2*x+1)^(2*n+1)),n,0,inf);
   simple(s1)  % 对结果进行化简，MATLAB 6.5 及以前版本因本身 bug 化简很麻烦
ans =
   log((x+1)/x)
```

【例 3-25】试求解级数与极限综合问题

$$\lim_{n\to\infty}\left[\left(1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n}\right)-\ln n\right]$$

【求解】前面介绍了级数求和，还介绍了极限的求解方法，所以这里给出的综合问题仍然能用 MATLAB 语言的符号运算工具箱直接求解。从题中给出的式子可见，其中包含级数求和项可以表示成 symsum(1/m,1,n)，这样原始的问题可以直接由下面的 MATLAB 语句求解。

```
>> syms m n; limit(symsum(1/m,m,1,n)-log(n),n,inf)
ans =
   eulergamma
```

亦即得出的结果为 Euler 常数 γ，其值可以由 MATLAB 精确地显示出来。

```
>> vpa(ans, 70)  % 显示 70 位有效数字
ans =
   0.5772156649015328606065120900824024310421593359399235988057672348848677
```

注意，求解该问题不能先求解无穷级数的和，然后再减去 $\ln n$，再求极限，这样做前后两项均为无穷大，求极限的结果将是不定式 NaN。

3.3 数值微分

前面介绍了已知原型函数，可以通过 diff() 函数求取各阶导数解析解的方法，并得出结论，高达 100 阶的导数也可以用 MATLAB 语言在几秒钟的时间内直接求出。应该指出，前面介绍的解析解方法的前提是原型函数为已知的。如果函数表达式未知，只有实验数据，在实际应用中经常也有求导的要求，这样的问题就不能用前面的方法获得问题的解析解。要求解这样的问题，需要引入数值算法得出所需问题的解。由于在 MATLAB

语言中没有现成的数值微分函数，所以本节将先介绍数值微分算法，介绍其中较好算法的 MATLAB 实现，最后将通过例子演示数值微分程序。

3.3.1 数值微分算法

假设已经等间隔地测出了一组数据 (t_i, y_i)，且已知时间间隔为 Δt，由高等数学中导数的定义可知，若 $\Delta t \to 0$，则相邻两点的差值除以间隔 Δt 就是该点处的导数，由此可以引入前向微分公式

$$y_i' = \frac{\Delta y_i}{\Delta t} = \frac{y_{i+1} - y_i}{\Delta t} \tag{3-3-1}$$

类似地，还可以引入后向差分公式

$$y_i' = \frac{\Delta y_i}{\Delta t} = \frac{y_i - y_{i-1}}{\Delta t} \tag{3-3-2}$$

遗憾的是，这两种微分算法的精度都是 $o(\Delta t)$ 级的，当 Δt 稍大时，产生的误差会很大。经实践检验，利用基于前向和后向差分的数值微分算法求取高阶微分时的精度一般都是很低的，所以这里只介绍两种中心差分的算法。首先定义一阶微分为

$$y_i' = \frac{\Delta y_i}{\Delta t} = \frac{y_{i+1} - y_{i-1}}{2\Delta t} \tag{3-3-3}$$

记

$$\widetilde{f}'(x) = \frac{f(x+\Delta t) - f(x-\Delta t)}{2\Delta t} \tag{3-3-4}$$

由 Taylor 级数展开可以将上式进一步写成

$$\begin{aligned}\widetilde{f}(x) = {} & \frac{f(x) + \Delta t f'(x) + \Delta t^2 f''(x)/2! + \Delta t^3 f''(\xi)/3! + o(\Delta t^4)}{2\Delta t} - \\ & \frac{f(x) - \Delta t f'(x) + \Delta t^2 f''(x)/2! - \Delta t^3 f''(\xi)/3! + o(\Delta t^4)}{2\Delta t} = f'(x) + \frac{\Delta t^3}{3!} f''(\xi)\end{aligned} \tag{3-3-5}$$

可见这种中心差分的算法精度为 $o(\Delta t^2)$。该中心差分算法的高阶微分公式为

$$\begin{aligned} y_i'' &= \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta t^2} \\ y_i''' &= \frac{y_{i+2} - 2y_{i+1} + 2y_{i-1} - y_{i-2}}{2\Delta t^3} \\ y_i^{(4)} &= \frac{y_{i+2} - 4y_{i+1} + 6y_i - 4y_{i-1} + y_{i-2}}{\Delta t^4} \end{aligned} \tag{3-3-6}$$

另一种具有 $o(\Delta t^4)$ 精度级的中心差分算法为

$$\begin{aligned} y_i' &= \frac{-y_{i+2} + 8y_{i+1} - 8y_{i-1} + y_{i-2}}{12\Delta t} \\ y_i'' &= \frac{-y_{i+2} + 16y_{i+1} - 30y_i + 16y_{i-1} - y_{i-2}}{12\Delta t^2} \\ y_i''' &= \frac{-y_{i+3} + 8y_{i+2} - 13y_{i+1} + 13y_{i-1} - 8y_{i-2} + y_{i-3}}{8\Delta t^3} \\ y_i^{(4)} &= \frac{-y_{i+3} + 12y_{i+2} - 39y_{i+1} + 56y_i - 39y_{i-1} + 12y_{i-2} - y_{i-3}}{6\Delta t^4} \end{aligned} \tag{3-3-7}$$

3.3.2 中心差分方法及其 MATLAB 实现

从前面的介绍可知，式 (3-3-7) 中给出的微分算法有 $o(\Delta t^4)$ 级精度，因而即使 Δt 不趋于 0 时，仍能得出较好的近似微分。所以这里采用该公式为所选算法，可以编写出一个 MATLAB 函数，其内容为

```
function [dy,dx]=diff_ctr(y, Dt, n)
yx1=[y 0 0 0 0 0]; yx2=[0 y 0 0 0 0]; yx3=[0 0 y 0 0 0];
yx4=[0 0 0 y 0 0]; yx5=[0 0 0 0 y 0]; yx6=[0 0 0 0 0 y];
switch n
case 1
   dy = (-diff(yx1)+7*diff(yx2)+7*diff(yx3)-diff(yx4))/(12*Dt); L0=3;
case 2
   dy=(-diff(yx1)+15*diff(yx2)-15*diff(yx3)+diff(yx4))/(12*Dt^2);L0=3;
case 3
   dy=(-diff(yx1)+7*diff(yx2)-6*diff(yx3)-6*diff(yx4)+...
         7*diff(yx5)-diff(yx6))/(8*Dt^3); L0=5;
case 4
   dy = (-diff(yx1)+11*diff(yx2)-28*diff(yx3)+28*diff(yx4)-...
      11*diff(yx5)+diff(yx6))/(6*Dt^4);L0=5;
end
dy=dy(L0+1:end-L0); dx=([1:length(dy)]+L0-2-(n>2))*Dt;
```

这样编写的 M-函数调用格式为

$[\boldsymbol{d}_y, \boldsymbol{d}_x]$=diff_ctr($\boldsymbol{y},\Delta t, n$)

其中，$\boldsymbol{y}$ 为给定的等间距的实测数据构成的向量，Δt 为自变量的间距，n 为所需的导数阶次。向量 $\boldsymbol{d}_y$ 为得出的导数向量，而 $\boldsymbol{d}_x$ 为相应的自变量向量。注意这两个向量的长度比 $\boldsymbol{y}$ 短。

【例 3-26】这里仍采用例 3-4 中给出的函数 (这里给出函数原型是为了精度检验)。由于原型函数已知，所以可以求出导数的解析解，从中求出精确的值。试用数值微分求取原函数的 1~4 阶导数，并和解析解比较精度。

【求解】生成一个横坐标点组成的向量 $\boldsymbol{x}$。另外由已知的原型函数，可以立即得出函数各阶导数的解析解，并将已知的横坐标点代入，即可以得出各阶导数精确的数值解，可以用于对照。

```
>> h=0.05; x=0:h:pi; syms x1; y=sin(x1)/(x1^2+4*x1+3);
   yy1=diff(y); f1=subs(yy1,x1,x);    % 求各阶导数的解析解与对照数据
   yy2=diff(yy1); f2=subs(yy2,x1,x); yy3=diff(yy2); f3=subs(yy3,x1,x);
   yy4=diff(yy3); f4=subs(yy4,x1,x);
```

假设由该原型函数可以生成一些数据点 y_i，由这些点可以拟合出曲线的 1~4 阶导数。下面的语句可以通过数值的方法获得已知数据点处的各阶导数，还可以绘制出曲线，将数值导数和由解析解计算出来的导数在相同的坐标系下绘制出来，如图 3-6 所示。可以看出，由数值方法获得的

导数是很精确的，其误差从图上是看不出来的。

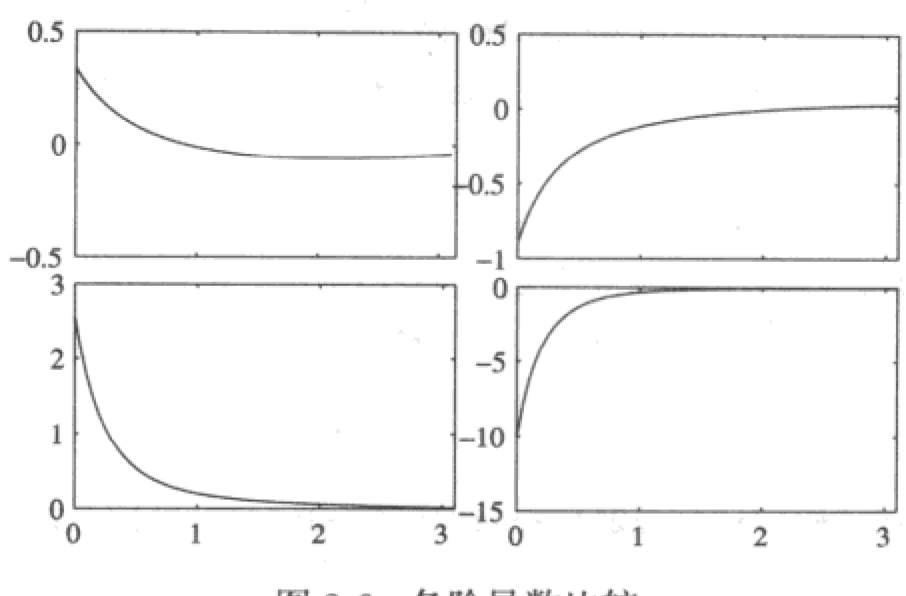


图 3-6 各阶导数比较

```
>> y=sin(x)./(x.^2+4*x+3);    % 生成已知数据点
   [y1,dx1]=diff_ctr(y,h,1); subplot(221), plot(x,f1,dx1,y1,':');
   [y2,dx2]=diff_ctr(y,h,2); subplot(222), plot(x,f2,dx2,y2,':')
   [y3,dx3]=diff_ctr(y,h,3); subplot(223), plot(x,f3,dx3,y3,':');
   [y4,dx4]=diff_ctr(y,h,4); subplot(224), plot(x,f4,dx4,y4,':')
```

下面定量地分析得出的误差，考虑计算得出的 4 阶导数向量，其长度比原始对照向量 f_4 短，所以两个向量取同样多点进行比较，就可以得出数值方法的相对误差最大值为 3.5×10^{-4}，亦即 0.035%。由此可见，这里的数值方法还是很精确的。

```
>> norm((y4-f4(4:60))./f4(4:60))
ans =
    3.502529066338957e-004
```

3.3.3 二元函数的梯度计算

如果给定二元函数的函数值矩阵 z，其中 z 为网格数据，则可以由 gradient() 函数求取二元函数的梯度。该函数的调用格式为

$[f_x,f_y]$=gradient(z)

其实，这样计算出来 f_x 与 f_y 不是真正的梯度。这里尚未考虑 x, y 坐标的情况。如果得到的 z 矩阵是建立在等间距的形式生成网格基础上的，则实际的梯度值可以如下求出。

f_x=$f_x/\Delta x$, f_y=$f_y/\Delta y$

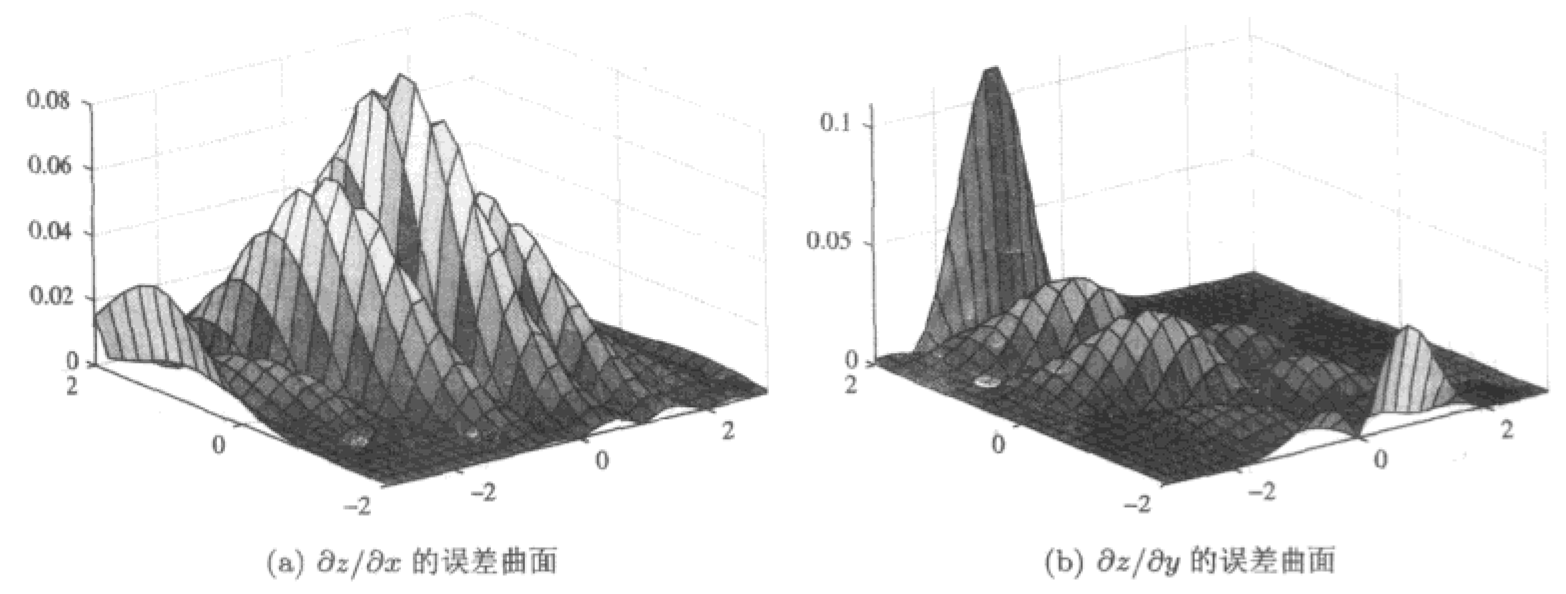
其中，Δx 和 Δy 分别为 x, y 生成网格的步距。

【例 3-27】考虑例 3-5 中给出的问题，假设已经得出网格数据，试用数值方法由该数据解出梯度值，并研究得出结果的误差。

【求解】现在重新生成数据，则可以由这些数据直接计算出该函数的梯度，而无需再从原函数直接计算梯度值。由下面的语句还将直接绘制出带有等值线的引力线图，和图 3-3 (b) 中的图形完全一致。

```
>> [x,y]=meshgrid(-3:.2:3,-2:.2:2); z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
   [fx,fy]=gradient(z); fx=fx/0.2; fy=fy/0.2;
   contour(x,y,z,30); hold on; quiver(x,y,fx,fy)
```

下面的语句将绘制出误差的曲面，如图 3-7 所示。可见大部分区域内误差还是较小的，但在某些小的区域内误差较大，这说明原来网格的间距较大，使得简单的梯度函数难以精确求解。

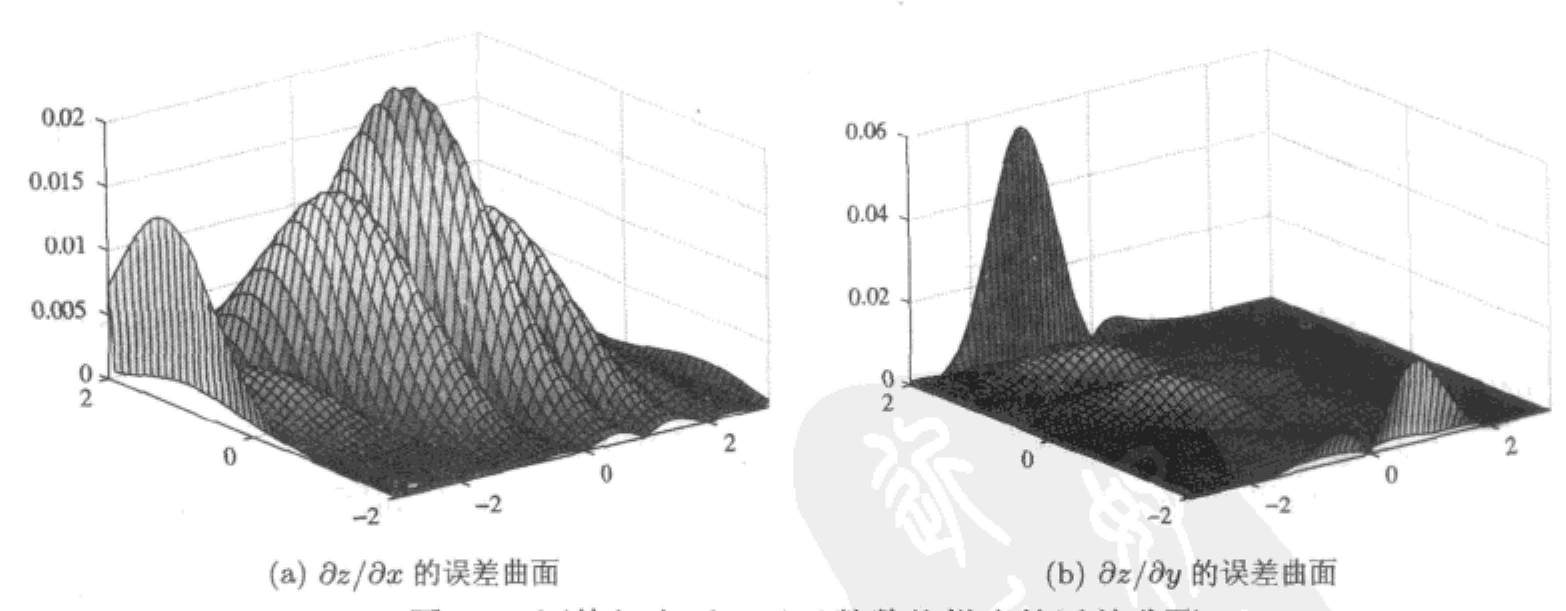


(a) $\partial z/\partial x$ 的误差曲面　　(b) $\partial z/\partial y$ 的误差曲面

图 3-7　二元函数数值梯度的误差曲面

```
>> zx=-exp(-x.^2-y.^2-x.*y).*(-2*x+2+2*x.^3+x.^2.*y-4*x.^2-2*x.*y);
   zy=-x.*(x-2).*(2*y+x).*exp(-x.^2-y.^2-x.*y);
   surf(x,y,abs(fx-zx)); axis([-3 3 -2 2 0,0.08])
   figure; surf(x,y,abs(fy-zy)); axis([-3 3 -2 2 0,0.11])
```

如果将网格加密一倍，则可以由下面的语句计算数值梯度，得出的结果与理论值求取误差则可以绘制出来，如图 3-8 所示，可见这时误差显著减小。



(a) $\partial z/\partial x$ 的误差曲面　　(b) $\partial z/\partial y$ 的误差曲面

图 3-8　网格加密后二元函数数值梯度的误差曲面

```
>> [x,y]=meshgrid(-3:.1:3,-2:.1:2); z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
   [fx,fy]=gradient(z); fx=fx/0.1; fy=fy/0.1;
   zx=-exp(-x.^2-y.^2-x.*y).*(-2*x+2+2*x.^3+x.^2.*y-4*x.^2-2*x.*y);
   zy=-x.*(x-2).*(2*y+x).*exp(-x.^2-y.^2-x.*y);
```

```
surf(x,y,abs(fx-zx)); axis([-3 3 -2 2 0,0.02])
figure; surf(x,y,abs(fy-zy)); axis([-3 3 -2 2 0,0.06])
```

3.4 数值积分问题

3.4.1 由给定数据进行梯形求积

一元函数定积分的数学表示为

$$I=\int_a^b f(x)\mathrm{d}x \tag{3-4-1}$$

在被积函数 $f(x)$ 理论上不可积时，即使有强大的计算机数学语言帮忙，也不能够求出该积分的解析解，所以往往要采用数值方法来求解。求解定积分的数值方法是多种多样的，如简单的梯形法、Simpson 法、Romberg 法等算法都是数值分析课程中经常介绍的方法。它们的基本思想都是将整个积分空间 $[a,b]$ 分割成若干个子空间 $[x_i,x_{i+1}]$，$i=1,2,\cdots,N$，其中 $x_1=a$，$x_{N+1}=b$。这样整个积分问题就分解为下面的求和形式。

$$\int_a^b f(x)\mathrm{d}x=\sum_{i=1}^N\int_{x_i}^{x_{i+1}} f(x)\mathrm{d}x=\sum_{i=1}^N \Delta f_i \tag{3-4-2}$$

而在每一个小的子空间上都可以近似地求解出来，当然最简单的求每一小的子空间的积分方法是采用梯形近似的方法。梯形方法还可以应用于已知数据样本点的数值积分问题求解。假设在实验中测得一组数据 $(x_1,y_1),(x_2,y_2),(x_3,y_3),\cdots,(x_N,y_N)$，且 x_i 为严格单调递增的数值，直接求取这些点对应曲线的数值积分最直观的方法就是用梯形方法，用直线将这些点连接起来，则积分可以近似为该折线与 x-轴之间围成的面积。这样，

$$S=\frac{1}{2}\left[\sum_{i=1}^{N-1}(y_{i+1}+y_i)(x_{i+1}-x_i)\right]=\frac{1}{2}\left\{\sum_{i=1}^{N-1}[(y_{i+1}-y_i)+2y_i](x_{i+1}-x_i)\right\} \tag{3-4-3}$$

由此可见，用 MATLAB 语言实现梯形积分算法很容易。

假设已经建立起向量 $\boldsymbol{x}=[x_1,x_2,\cdots,x_N]^\mathrm{T}$，$\boldsymbol{y}=[y_1,y_2,\cdots,y_N]^\mathrm{T}$，则可以用下面的语句得出该积分的值为

```
sum((2*y(1:end-1,:)+diff(y)).*diff(x))/2
```

MATLAB 提供的 `trapz()` 函数也可以直接用梯形法求解积分问题，该函数调用格式为

```
S=trapz(x,y)
```

其中，$\boldsymbol{x}$ 可以为行向量或列向量，$\boldsymbol{y}$ 的行数应该等于 $\boldsymbol{x}$ 向量的元素数。如果 $\boldsymbol{y}$ 由多列矩阵给出，则用该函数可以得出若干个函数的积分值。

【例 3-28】 试用梯形法求出 $x\in(0,\pi)$ 区间内，函数 $\sin(x),\cos(x),\sin(x/2)$ 的定积分值。

【求解】 生成区间内横坐标向量，用上述的算法可以求出各个函数的数值积分值。

```
>> x1=[0:pi/30:pi]'; y=[sin(x1) cos(x1) sin(x1/2)];
   x=[x1 x1 x1]; S=sum((2*y(1:end-1,:)+diff(y)).*diff(x))/2
S =
   1.99817196134365   0.00000000000000   1.99954305299081
>> S1=trapz(x1,y)    % 得出和上述完全一致的结果
```

由于选择的步距较大，为 $h=\pi/30\simeq 0.1$，故得出的结果有较大的误差。在第 8.1.2 节中将积分问题与样条插值技术相结合，给出一个能精确计算积分的 MATLAB 函数，并演示其在更大步距下的有效性。

【例 3-29】请用定步长方法求解积分 $\int_0^{3\pi/2}\cos(15x)\mathrm{d}x$。

【求解】求解问题之前，首先用下面的 MATLAB 语句绘制出被积函数的曲线，如图 3-9 所示。可见，在求解区域内被积函数有很强的振荡。

```
>> x=[0:0.01:3*pi/2, 3*pi/2];  % 这样赋值能确保 3π/2 点被包含在内
   y=cos(15*x); plot(x,y)
```

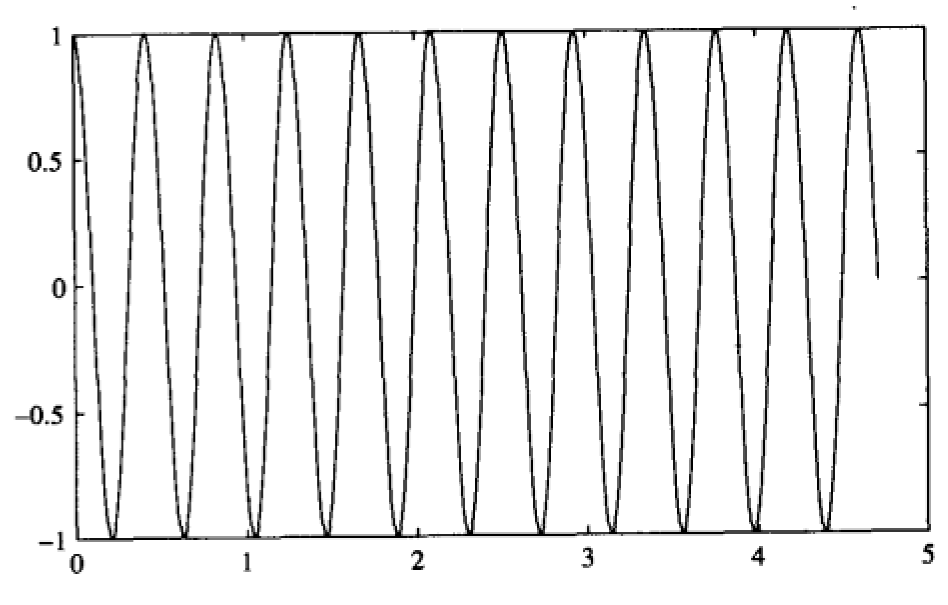


图 3-9 被积函数 $f(x)=\cos(15x)$ 的曲线

对不同的步距 $h=0.1,0.01,0.001,0.0001,0.00001,0.000001$，可以用下面的语句求出采用不同步距的积分近似结果。为了有更好的表示，特将结果用表 3-1 列出。

```
>> syms x, A=int(cos(15*x),0,3*pi/2)   % 求取理论值
A =
  1/15
>> h0=[0.1,0.01,0.001,0.0001,0.00001,0.000001]; v=[];
   for h=h0,
      x=[0:h:3*pi/2, 3*pi/2]; y=cos(15*x); I=trapz(x,y); v=[v; h,I,1/15-I];
   end
```

可见，随着步距 h 的减小，计算精度逐渐增加。例如，当 $h=10^{-6}$ 时可以保留小数点后 11 位精确数字，但这时求解的时间也显著增加，达到半分钟，如果想进一步增加计算精度，还得再减小步长，这样计算时间会增加到难以接受的值。

表 3-1 步距选择与计算结果

步长	得出积分值	误差	步长	得出积分值	误差
0.1	0.05389175150075948	0.0127749152	0.0001	0.0666665416666881	1.24999978×10^{-8}
0.01	0.0665416954658383	0.0001249712	10^{-5}	0.0666666654166685	1.24999816×10^{-10}
0.001	0.06666541668003727	1.2499866×10^{-6}	10^{-6}	0.0666666666541621	1.25045807×10^{-12}

3.4.2 单变量数值积分问题求解

单变量函数的数值积分还可以采用一般数值分析中介绍的其他算法进行求解。例如，可以采用下面给出的 Simpson 方法来求解出 $[x_i,x_{i+1}]$ 上的积分 Δf_i 的近似值为

$$\Delta f_i\simeq\frac{h_i}{12}\left[f(x_i)+4f\left(x_i+\frac{h_i}{4}\right)+2f\left(x_i+\frac{h_i}{2}\right)+4f\left(x_i+\frac{3h_i}{4}\right)+f(x_i+h_i)\right] \tag{3-4-4}$$

式中，$h_i=x_{i+1}-x_i$。MATLAB 基于此算法，采用自适应变步长方法给出了 quad() 函数来求取定积分，该函数的调用格式为

y=quad(Fun,a,b)	求定积分
y=quad(Fun,a,b,ϵ)	限定精度的定积分求解

其中，Fun 为描述被积函数的字符串变量，可以是一个 Fun.m 函数文件名，该函数的一般格式为 y= Fun(x)，还可以用 inline() 函数直接定义。a,b 分别为定积分的上限和下限，ϵ 为用户指定的误差限，默认值为 10^{-6}。给定了这些因素，可以利用 MATLAB 的求积分函数 quad() 直接求解定积分问题的数值解。下面将通过例子演示数值积分的求解方法。

【例 3-30】考虑前面提及的数学函数 $\mathrm{erf}(x)=\dfrac{2}{\sqrt{\pi}}\displaystyle\int_0^x \mathrm{e}^{-t^2}\mathrm{d}t$，由于其解析不可积，故需要用数值方法来求解。

【求解】 在求取数值解之前，需要描述一下被积函数。描述被积函数有两种方法，其一是建立一个 MATLAB 函数并将其存成文件，其内容为

```
function y=c3ffun(x)
y=2/sqrt(pi)*exp(-x.^2);
```

这样，可以将上述内容存入一个 c3ffun.m 文件。另一种描述被积函数的方式是用 inline() 函数定义被积函数，可以给出下面的语句。

```
>> f=inline('2/sqrt(pi)*exp(-x.^2)','x');
```

这样的方法无需建立一个单独的 MATLAB 文件。inline() 函数的第一个输入变量为被积函数本身，和 MATLAB 函数描述格式完全相同，第 2 个输入变量为自变量，当然还可以带有多个自变量。

定义了被积函数，就可以调用 quad() 函数直接求解出定积分的值。

```
>> f=inline('2/sqrt(pi)*exp(-x.^2)','x');
   y=quad(f,0,1.5)    % 用 inline 函数定义被积函数
```

```
y =
    0.96610518623173
>> y=quad('c3ffun',0,1.5)  % 用 M-函数定义被积函数
y =
    0.96610518623173
```

用两种方法得出的结果完全相同。其实，用符号运算工具箱可以求解出更精确的解如下:

```
>> syms x, y0=vpa(int(2/sqrt(pi)*exp(-x^2),0,1.5),60)
y0 =
   .966105146475310713936933729949905794996224943257461473285749
```

比较精确解和前面得出的数值解可见，数值解精度不高，用户可以试着减小误差限 tol 值的方法获取更高精度的解。

```
>> y=quad(f,0,1.5,1e-20)     % 设置高精度，但该方法失效
Warning: Maximum function count exceeded; singularity likely.
(Type "warning off MATLAB:quad:MaxFcnCount" to suppress this warning.)
   > In quad at 88
y =
   0.96606026001535
```

MATLAB 还提供了一个新的函数 quadl()，其调用的格式和 quad() 完全一致，使用的算法是 Lobbato 算法，其精度和速度均远高于 quad() 函数，所以在追求高精度数值解时可以采用这个方法。

早期版本的 MATLAB 还实现了 8 阶 Newton-Cotes 算法，给出了可用的积分函数 quad8()，其调用格式与 quad() 完全一致，精度和速度均优于 quad()，在目前的版本下一般不再建议使用该函数，而建议统一使用 quadl()。

【例 3-31】仍然考虑上面的问题，试提高求解的精度。

【求解】 使用 quadl() 函数可以立即得出如下结果。和精确解相比，其误差达到 10^{-16} 级，虽然未达到预期的 10^{-20} 级，但已经足够精确了。事实上，毕竟数值解采用的是双精度变量，所以精度也不可能达到 10^{-20} 级。

```
>> y=quadl(f,0,1.5,1e-20)
y =
    0.96610514647531
>> abs(y-y0)
ans =
   .6402522848913892e-16
```

【例 3-32】求解下面分段函数的积分问题。

$$I=\int_0^4 f(x)\mathrm{d}x,\ 其中\ f(x)=\begin{cases} \mathrm{e}^{x^2}, & 0\leqslant x\leqslant 2 \\ \dfrac{80}{4-\sin(16\pi x)}, & 2<x\leqslant 4 \end{cases}$$

【求解】 用曲线绘制函数不难绘制出分段函数，这里为减小视觉上的误差，在端点和间断点处采用了特殊处理，故可以得出如图 3-10 所示的填充图形。可见，在 $x=2$ 点处有跳跃。

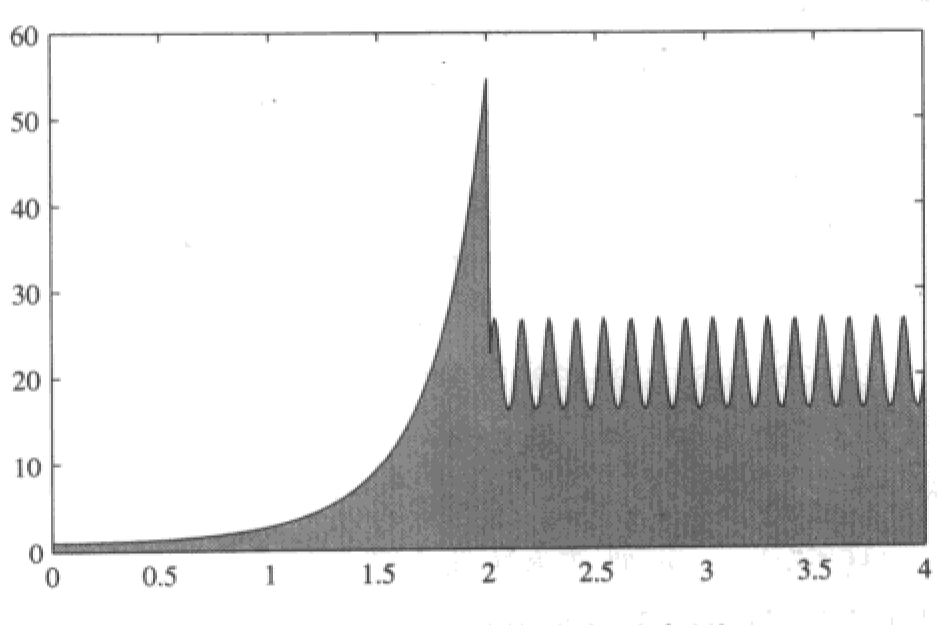


图 3-10 被积区域填充示意图

```
>> x=[0:0.01:2, 2+eps:0.01:4,4];
   y=exp(x.^2).*(x<=2)+80./(4-sin(16*pi*x)).*(x>2);
   y(end)=0; x=[eps, x]; y=[0,y]; fill(x,y,'g')
```

利用关系表达式可以描述出被积函数，再分别调用两种积分函数 quad() 和 quadl() 就可以求解出原始问题了。

```
>> f=inline('exp(x.^2).*(x<=2)+80*(x>2)./(4-sin(16*pi*x))','x');
   I1=quad(f,0,4)
I1 =
    57.76435412500863
>> I2=quadl(f,0,4)
I2 =
    57.76445016946768
```

不过从得出的结果看，二者有很大的差异。其实，可以将原来的积分问题转换成 $\int_0^2+\int_2^4$ 的问题，用积分问题解析解求解函数 int() 可以得出原始问题的解析解。

```
>> syms x; I=vpa(int(exp(x^2),0,2)+int(80/(4-sin(16*pi*x)),2,4))
I =
   57.764450125053010333315235385182
```

与解析解对比可见，用 quad() 函数得出的积分误差相当大。如果将积分区域分成两个部分单独计算，仍将得出很大的误差。由此可见该函数的局限性。在实际积分运算中应该慎用或不用该函数。但采用 quadl() 函数由分段函数积分将明显改善精度，若人为指定误差限则能得出更精确的结果。

```
>> f1=inline('exp(x.^2)','x'); f2=inline('80./(4-sin(16*pi*x))','x');
   quad(f1,0,2)+quad(f2,2,4)
ans =
```

```
    57.76444288914186
>> quadl(f1,0,2)+quadl(f2,2,4)
ans =
    57.76445012538125
>> quadl(f1,0,2,1e-11)+quadl(f2,2,4,1e-11)  % 人为给定精度限制
ans =
    57.76445012505302
```

【例 3-33】 试用 quad() 和 quadl() 函数分别求解例 3-29 中的定积分问题。

【求解】 从例 3-29 中演示的定步长方法看，只有步长选得极小，才能得出小数点后 11 位有效数字，且耗时较长。其实，用变步长数值积分函数可以轻而易举地求出该定积分问题的解，且使用的时间大大减少。

```
>> f=inline('cos(15*x)','x');
   tic, S=quadl(f,0,3*pi/2,1e-15), toc
S =
   0.06666666666667
Elapsed time is 1.973000 seconds.
```

所以，由此可以得出结论。求解变化不均匀的函数的积分不宜采用传统数值分析类课程介绍的定步长积分算法，因为用该算法精度难以保证；而若要使用小步长，则计算量将极大，且仍然无法保证计算精度。采用变步长算法可以很容易地得出原问题的解。

同样考虑用 quad() 函数求解该问题，则可以给出如下语句：

```
>> S1=quad(f,0,3*pi/2)    % 采用默认精度
S1 =
   0.06666665694139
```

可见，这样计算的结果精度不高，基本相当于 $h=0.00001$ 时的梯形法结果，不是很令人满意。仿照 quadl() 函数，也给出 10^{-15} 的精度要求，则无法得到收敛的结果。所以对一般定积分求取问题，数值方法的首选为 quadl() 函数。

```
>> S1=quad(f,0,3*pi/2,1e-15)
Warning: Maximum function count exceeded; singularity likely.
(Type "warning off MATLAB:quad:MaxFcnCount" to suppress this warning.)
> In quad at 88
S1 =
   1.27830264040033
```

3.4.3 双重积分问题的数值解

考虑下面的双重定积分问题。

$$I=\int_{y_m}^{y_M}\int_{x_m}^{x_M}f(x,y)\mathrm{d}x\mathrm{d}y \tag{3-4-5}$$

使用 MATLAB 提供的 dblquad() 函数就可以直接求出上述双重定积分的数值解。该函数的调用格式为

y=dblquad(Fun, x_m, x_M, y_m, y_M)　　矩形区域的双重积分

y=dblquad(Fun, x_m, x_M, y_m, y_M, ϵ)　限定精度的双重积分

注意，本函数不允许返回被积函数调用次数，故用户可以自己在被积函数中设置一个计数器，从而测出调用次数。

【例 3-34】 试求出双重定积分

$$J=\int_{-1}^{1}\int_{-2}^{2}\mathrm{e}^{-x^2/2}\sin(x^2+y)\mathrm{d}x\mathrm{d}y$$

【求解】 由给出的被积函数可以简单地写出下面的函数，这样就可以通过下面的MATLAB语句求出被积函数的双重定积分了。

```
>> f=inline('exp(-x.^2/2).*sin(x.^2+y)','x','y');
   y=dblquad(f,-2,2,-1,1),
y =
     1.57456866245358
```

遗憾的是，在 MATLAB 中并没有提供求解更一般的双重积分问题的函数

$$I=\int_{x_m}^{x_M}\int_{y_m(x)}^{y_M(x)}f(x,y)\mathrm{d}y\mathrm{d}x \tag{3-4-6}$$

好在美国学者 Howard Wilson 与 Bryce Gardner 开发了数值积分工具箱 (NIT，即 Numerical Integration Toolbox)。该工具箱可以在 The MathWorks 公司的网站上免费下载[①]，该工具箱中的函数 gquad2dggen() 可以直接求解式 (3-4-6) 的双重积分问题。该函数的调用格式为

J=quad2dggen(Fun,$\mathrm{F_{lower}}$,$\mathrm{F_{upper}}$,y_m,y_M)　　一般双重积分

J=quad2dggen(Fun,$\mathrm{F_{lower}}$,$\mathrm{F_{upper}}$,y_m,y_M,ϵ)　限定精度的双重积分

其中，ϵ 为误差限。误差限越小，计算应该越精确，但计算量也将增大，此函数默认的误差限为 $\epsilon=10^{-3}$。该函数还涉及 3 个 MATLAB 函数，即被积函数和上下限函数。

注意，该函数指定的积分顺序是先 x 后 y，若想求先 y 后 x 的积分，则需要交换被积函数中 x, y 顺序。下面将通过一个具体例子来演示双重积分的运算。

【例 3-35】 试求出双重定积分

$$J=\int_{-1/2}^{1}\int_{-\sqrt{1-x^2/2}}^{\sqrt{1-x^2/2}}\mathrm{e}^{-x^2/2}\sin(x^2+y)\mathrm{d}y\mathrm{d}x$$

[①]也可以从作者维护的“MATLAB 大观园”中下载。目前该软件是在 MATLAB 4.2 下调试的，在 6.x 版本下可能会有不兼容之处，但数值计算部分的内核不会有问题。另外，安装完该工具箱后，别忘了将其路径加载到 MATLAB 路径下。

【求解】 这里的例子是先 x 后 y，可以先构造出 $x_M(y)$ 和 $x_m(y)$ 函数，再调用相应的函数求解原问题。注意，这里应该首先交换积分变量次序。

```
>> fh=inline('sqrt(1-x.^2/2)','x');  % 内积分上限
   fl=inline('-sqrt(1-x.^2/2)','x'); % 内积分下限
   f=inline('exp(-x.^2/2).*sin(x.^2+y)','y','x');  % 交换顺序的被积函数
   y=quad2dggen(f,fl,fh,-1/2,1,eps),
y =
     0.41192954617630
>> syms x y  % 现在考虑解析解方法
   i1=int(exp(-x^2/2)*sin(x^2+y),y,-sqrt(1-x^2/2),sqrt(1-x^2/2));
   int(i1,x,-1/2,1) % 求取解析解时得出错误信息
   vpa(ans)
Warning: Explicit integral could not be found.
> In sym.int at 58
ans =
   int(2*exp(-1/2*x^2)*sin(x^2)*sin(1/2*(4-2*x^2)^(1/2)),x = -1/2 .. 1)
>> vpa(ans,70)   % 不存在解析解，但可以求高精度数值解
ans =
   .4119295461762951196517599401760134672761827128252391627415959533602675
```

可见，前面由 quadl() 函数得出数值解也是很精确的。本问题可以求出高精度数值解的主要原因是被积函数的内积分是可积的。如果不可积，则无从得出整个双重积分，这样只能采用数值解。例如，若积分问题变成

$$J=\int_{-1}^{1}\int_{-\sqrt{1-y^2}}^{\sqrt{1-y^2}}\mathrm{e}^{-x^2/2}\sin(x^2+y)\mathrm{d}x\mathrm{d}y$$

则对 x 是不可积的，故调用解析解方法不会得出结果，而数值解求解不受此影响。

```
>> fh=inline('sqrt(1-y.^2)','y');  % 内积分上限
   fl=inline('-sqrt(1-y.^2)','y'); % 内积分下限
   f=inline('exp(-x.^2/2).*sin(x.^2+y)','x','y');  % 交换顺序的被积函数
   I=quad2dggen(f,fl,fh,-1,1,eps),
I =
   0.53686038269795
```

3.4.4 三重定积分的数值求解

长方体区域的三重定积分

$$I=\int_{x_m}^{x_M}\int_{y_m}^{y_M}\int_{z_m}^{z_M}f(x,y,z)\mathrm{d}z\mathrm{d}y\mathrm{d}x \tag{3-4-7}$$

可以由 MATLAB 提供的 triplequad() 函数得出。该函数调用格式为

I=triplequad(Fun,x_m,x_M,y_m,y_M,z_m,z_M,ϵ,@quadl)

其中，Fun 描述三元被积函数，可以用 M-函数或 inline() 函数定义。ϵ 变量是积分精度控制量，默认值为 10^{-6}，为提高精度起见，还可以选择更小的值。@quadl 为具体求解一元积分的数值函数，也可以选择为 @quad 甚至是用户自己编写的积分函数，但要求其调用格式与 quadl() 函数完全一致。

【例 3-36】用数值方法求解例 3-16 中的三重定积分问题

$$\int_0^1\int_0^\pi\int_0^\pi 4xz\mathrm{e}^{-x^2y-z^2}\mathrm{d}z\mathrm{d}y\mathrm{d}x$$

【求解】用 inline() 函数可以描述被积函数，通过下面的语句立即可以求出三重定积分值。

```
>> triplequad(inline('4*x.*z.*exp(-x.*x.*y-z.*z)','x','y','z'),...
        0, 2, 0, pi, 0, pi,1e-7,@quadl)
ans =
     3.10807945143834
```

NIT 工具箱还可以解决多重超维长方体边界的定积分问题。例如，使用 quadndg() 函数，但对一般积分区域来说则没有现成的求解函数。另外，该工具箱单重积分函数 quadg() 的调用格式和 quad() 一致，其效率也高于 quadl()，故在进行数值求积分时建议使用此工具箱。

3.5 曲线积分与曲面积分的计算

MATLAB 语言和 Maple 等计算机数学语言并未直接提供曲线积分和曲面积分的现成函数。本节将介绍两类曲线、曲面积分的概念，引入它们转换成一般积分问题的算法，并介绍利用 MATLAB 语言的符号运算工具箱直接求解曲线、曲面积分的解析解方法。

3.5.1 曲线积分及 MATLAB 求解

3.5.1.1 第一类曲线积分

曲线积分在高等数学中一般分为第一类曲线积分和第二类曲线积分。其中，第一类曲线积分问题起源于对不均匀分布的空间曲线总质量的求取[3]。假设在空间曲线 l 上的密度函数为 $f(x,y,z)$，则其总质量，亦即第一类曲线积分的值可以由下面的式子直接求出

$$I_1=\int_l f(x,y,z)\mathrm{d}s \tag{3-5-1}$$

其中，s 为曲线上某点的弧长，所以这类曲线积分又称为对弧长的曲线积分。若 x,y,z 均由参数方程 $x=x(t),y=y(t),z=z(t)$ 给出，则可以将这些量直接调入 $f(\cdot)$ 函数，而弧长可以表示成

$$\mathrm{d}s=\sqrt{\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^2+\left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)^2+\left(\frac{\mathrm{d}z}{\mathrm{d}t}\right)^2}\mathrm{d}t\text{，简记作 }\mathrm{d}s=\sqrt{x_t^2+y_t^2+z_t^2}\mathrm{d}t \tag{3-5-2}$$

则可以将这类曲线积分也变换成对参数 t 的普通定积分问题

$$I=\int_{t_m}^{t_M} f[x(t),y(t),z(t)]\sqrt{x_t^2+y_t^2+z_t^2}\mathrm{d}t \tag{3-5-3}$$

若被积函数 $f(x,y)$ 为二元函数，也可以用相应的转换方法将其转换成普通积分问题，故用 MATLAB 语言可以求出第一类曲线积分的值。

【例 3-37】 试求 $\int_l \frac{z^2}{x^2+y^2}\mathrm{d}s$，其中 l 为螺线，$x=a\cos t, y=a\sin t, z=at, (0\leqslant t\leqslant 2\pi, a>0)$。

【求解】 用下面的语句可以立即得出曲线积分值。

```
>> syms t; syms a positive; x=a*cos(t); y=a*sin(t); z=a*t;
   I=int(z^2/(x^2+y^2)*sqrt(diff(x,t)^2+diff(y,t)^2+diff(z,t)^2),t,0,2*pi)
```

用 LaTeX 显示该结果得 $I=\frac{8\sqrt{2}}{3}\pi^3 a$。

【例 3-38】 试求 $\int_l (x^2+y^2)\mathrm{d}s$，其中 l 曲线为 $y=x$ 与 $y=x^2$ 围成的正向曲线。

【求解】 应该用下面的指令绘制出给定的两条曲线，如图 3-11 所示。

```
>> x=0:.001:1.2; y1=x; y2=x.^2; plot(x,y1,x,y2)
```

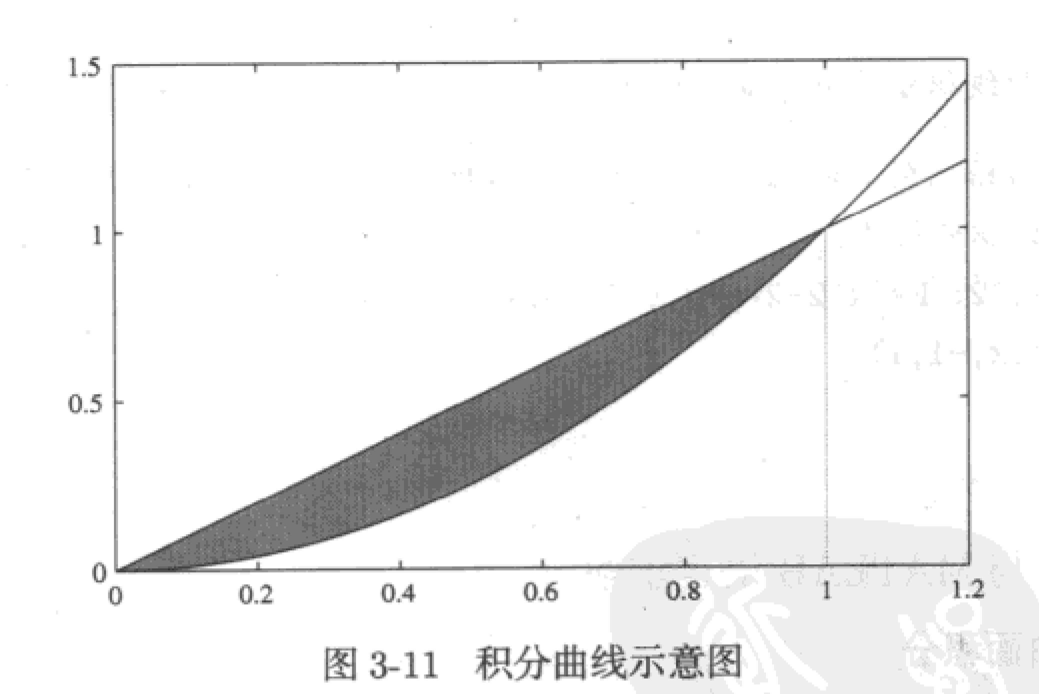


图 3-11 积分曲线示意图

可见，可以将原来的积分问题化成两段曲线的积分问题来求解。故应该给出如下的指令，求解出两段曲线的积分值，将其相加则得出原问题的解。

```
>> syms x; y1=x; y2=x^2; I1=int((x^2+y2^2)*sqrt(1+diff(y2,x)^2),x,0,1);
   I2=int((x^2+y1^2)*sqrt(1+diff(y1,x)^2),x,1,0); I=I2+I1
I =
   -2/3*2^(1/2)+349/768*5^(1/2)+7/512*log(-2+5^(1/2))
```

3.5.1.2 第二类曲线积分

第二类曲线积分问题又称为对坐标的曲线积分，它起源于变力 $\vec{f}(x,y,z)$ 沿曲线 l 移动时作功的研究。这类曲线积分的数学表达式为

$$I_2=\int_l \vec{f}(x,y,z)\cdot \mathrm{d}\vec{s} \tag{3-5-4}$$

其中，($\vec{f}(x,y,z)$ 为向量，可以写成 $\vec{f}=[P(x,y,z),Q(x,y,z),R(x,y,z)]$，曲线 $\mathrm{d}\vec{s}$ 亦为向量，若曲线可以由参数方程表示成 t 的函数，记作 $x(t),y(t),z(t)$，则可以将 $\mathrm{d}\vec{s}$ 表示成

$$\mathrm{d}\vec{s}=\left[\frac{\mathrm{d}x}{\mathrm{d}t},\frac{\mathrm{d}y}{\mathrm{d}t},\frac{\mathrm{d}z}{\mathrm{d}t}\right]^{\mathrm{T}}\mathrm{d}t \tag{3-5-5}$$

则两个向量的点乘可以由这两个向量直接得出，这样就可以利用 MATLAB 语言求出第二类曲线积分的值。

【例 3-39】 试求出曲线积分 $\int_l \frac{x+y}{x^2+y^2}\mathrm{d}x-\frac{x-y}{x^2+y^2}\mathrm{d}y$，$l$ 为正向圆周 $x^2+y^2=a^2$。

【求解】 若想按圆周曲线进行积分，则可以写出参数方程 $x=a\cos(t), y=a\sin(t), (0\leqslant t\leqslant 2\pi)$，这样，用下面的方法可以直接求出曲线积分。

```
>> syms t; syms a positive; x=a*cos(t); y=a*sin(t);
   F=[(x+y)/(x^2+y^2),-(x-y)/(x^2+y^2)]; ds=[diff(x,t);diff(y,t)];
   I=int(F*ds,t,2*pi,0)  % 正向圆周
I =
     2*pi
```

【例 3-40】 试求出曲线积分的值 $\int_l (x^2-2xy)\mathrm{d}x+(y^2-2xy)\mathrm{d}y$，$l$ 为抛物线 $y=x^2\ (-1\leqslant x\leqslant 1)$。

【求解】 其实，曲线给出的方程已经是关于 x 的参数方程，且 x 对 x 的导数显然为 1，故可以用下面的语句求出曲线积分的值。

```
>> syms x; y=x^2; F=[x^2-2*x*y,y^2-2*x*y]; ds=[1; diff(y,x)];
   I=int(F*ds,x,-1,1)
I =
   -14/15
```

3.5.2 曲面积分与 MATLAB 语言求解

3.5.2.1 第一类曲面积分

第一类曲面积分的数学定义为

$$I=\iint_S \phi(x,y,z)\mathrm{d}S \tag{3-5-6}$$

其中，$\mathrm{d}S$ 为小区域的面积，故这类积分又称为对面积的曲面积分。曲面 S 由 $z=f(x,y)$ 给出，则该积分可以转换成 x-y 平面的二重积分为

$$I=\iint_{\sigma_{xy}} \phi[x,y,f(x,y)]\sqrt{1+f_x^2+f_y^2}\mathrm{d}x\mathrm{d}y \tag{3-5-7}$$

其中，σ_{xy} 为积分区域。

【例 3-41】 试求出 $\iint_S xyz\mathrm{d}S$，其中积分曲面 S 为 $x=0,y=0,z=0$ 和 $x+y+z=a$ 围成，且 $a>0$。

【求解】 记这四个平面为 S_1,S_2,S_3,S_4，则原积分可以由 $\iint_S=\iint_{S_1}+\iint_{S_2}+\iint_{S_3}+\iint_{S_4}$ 求出。考虑 S_1,S_2,S_3 平面，由于被积函数的值为 0，故这些积分也为 0，所以只需研究 S_4 的曲线积分。S_4 平面的数学表示为 $z=a-x-y$，故由下面的语句可以求出曲面积分的结果。

```
>> syms x y; syms a positive; z=a-x-y;
   I=int(int(x*y*z*sqrt(1+diff(z,x)^2+diff(z,y)^2),y,0,a-x),x,0,a)
I =
   1/120*3^(1/2)*a^5
```

若曲面由参数方程

$$x=x(u,v),\ y=y(u,v),\ z=z(u,v) \tag{3-5-8}$$

给出，则曲面积分可以由下面的公式求出。

$$I=\iint_\Sigma \phi[x(u,v),y(u,v),z(u,v)]\sqrt{EG-F^2}\mathrm{d}u\mathrm{d}v \tag{3-5-9}$$

式中，

$$E=x_u^2+y_u^2+z_u^2,\ F=x_ux_v+y_uy_v+z_uz_v,\ G=x_v^2+y_v^2+z_v^2 \tag{3-5-10}$$

【例 3-42】 试求出曲面积分 $\iint(x^2y+zy^2)\mathrm{d}S$，其中 S 为螺旋曲面 $x=u\cos v,y=u\sin v,z=v$ 的 $0\leqslant u\leqslant a,0\leqslant v\leqslant 2\pi$ 部分。

【求解】 由说明介绍的公式可以立即得出积分结果。

```
>> syms u v; syms a positive;
   x=u*cos(v); y=u*sin(v); z=v; f=x^2*y+z*y^2;
   E=simple(diff(x,u)^2+diff(y,u)^2+diff(z,u)^2);
   F=diff(x,u)*diff(x,v)+diff(y,u)*diff(y,v)+diff(z,u)*diff(z,v);
   G=simple(diff(x,v)^2+diff(y,v)^2+diff(z,v)^2);
   I=int(int(f*sqrt(E*G-F^2),u,0,a),v,0,2*pi)
```

由 LaTeX 可以显示得出的结果为 $I=\dfrac{1}{8}\pi^2\left(2a\left(a^2+1\right)^{3/2}-a\sqrt{a^2+1}-\mathrm{arcsinh}a\right)$。

3.5.2.2 第二类曲面积分

第二类曲面积分又称为对坐标的曲面积分。其数学定义为

$$I=\iint_{S^+}P(x,y,z)\mathrm{d}y\mathrm{d}z+Q(x,y,z)\mathrm{d}x\mathrm{d}z+R(x,y,z)\mathrm{d}x\mathrm{d}y \tag{3-5-11}$$

其中，正向曲面 S^+ 由 $z=f(x,y)$ 给出，这类曲面积分问题可以转换成第一类曲面积分

$$I=\iint_{S^+}\left[P(x,y,z)\cos\alpha+Q(x,y,z)\cos\beta+R(x,y,z)\cos\gamma\right]\mathrm{d}S \tag{3-5-12}$$

其中，z 由 $f(x,y)$ 代替，且

$$\cos\alpha=\frac{-f_x}{\sqrt{1+f_x^2+f_y^2}},\quad \cos\beta=\frac{-f_y}{\sqrt{1+f_x^2+f_y^2}},\quad \cos\gamma=\frac{1}{\sqrt{1+f_x^2+f_y^2}} \tag{3-5-13}$$

这样，分母上的 $\sqrt{1+f_x^2+f_y^2}$ 正好和式 (3-5-7) 中的相应项抵消，故整个曲面积分可以写成

$$I=\iint_{\sigma_{xy}} -Pf_x\mathrm{d}x\mathrm{d}y-Qf_y\mathrm{d}x\mathrm{d}z+R\mathrm{d}y\mathrm{d}z \tag{3-5-14}$$

若曲面由参数方程 (3-5-8) 给出，则可以由下面的方程求出

$$\cos\alpha=\frac{A}{\sqrt{A^2+B^2+C^2}},\quad \cos\beta=\frac{B}{\sqrt{A^2+B^2+C^2}},\quad \cos\gamma=\frac{C}{\sqrt{A^2+B^2+C^2}} \tag{3-5-15}$$

其中，$A=y_uz_v-z_uy_v$，$B=z_ux_v-x_uz_v$，$C=x_uy_v-y_ux_v$。这样，由得出的第一类曲面积分转换成二重积分会发现，式 (3-5-15) 的分母正好和 $\sqrt{EG-F^2}$ 抵消。这时整个曲面积分可以简化成

$$I=\iint_{S^+}[AP(u,v)+BQ(u,v)+CR(u,v)]\mathrm{d}u\mathrm{d}v \tag{3-5-16}$$

【例 3-43】试求出曲面积分 $\iint x^3\mathrm{d}y\mathrm{d}z$，其中 S 是椭球面 $\dfrac{x^2}{a^2}+\dfrac{y^2}{b^2}+\dfrac{z^2}{c^2}=1$ 的上半部，且积分沿椭球面的上面。

【求解】可以引入参数方程 $x=a\sin u\cos v, y=b\sin u\sin v, z=c\cos u$，且 $\left(0\leqslant u\leqslant\dfrac{\pi}{2}\right)$，$(0\leqslant v\leqslant 2\pi)$，则可以用下面的语句求出所需的曲面积分。

```
>> syms u v; syms a b c positive;
   x=a*sin(u)*cos(v); y=b*sin(u)*sin(v); z=c*cos(u);
   A=diff(y,u)*diff(z,v)-diff(z,u)*diff(y,v);
   I=int(int(x^3*A,u,0,pi/2),v,0,2*pi)
I =
   2/5*pi*a^3*c*b
```

3.6 本章要点简介

- 本章介绍的函数和有关的微积分问题求解的函数由下表给出。

函数名	函 数 功 能	工具箱	本书页码
limit()	极限问题求解和单边极限求解问题，可嵌套求多变量极限	符号运算	47,48
diff()	求解导数问题，还可以用于求高阶导数和偏导数	符号运算	48,50
语句	隐函数偏导公式与求解，见式 (3-1-6)	除法	52

续表

函数名	函 数 功 能	工具箱	本书页码
语句	参数方程求导公式与求解	除法	53
int()	求解不定积分与定积分，可嵌套求解多重积分	符号运算	53,54
taylor()	Taylor 幂级数展开	符号运算	57
jacobian()	Jacobi 矩阵求解	符号运算	52
mtaylor	多变量函数的 Taylor 展开	Maple	59
fseriies()	Fourier 级数展开，或采用定义直接积分，求解级数系数	符号运算	60
symsum()	级数求和，可以用于无穷级数的求和	符号运算	61
diff_ctr()	中心差分算法求解 1~4 阶导数，见式 (3-3-7)	自编	65
gradient()	二元函数的梯度计算，真正的梯度还应该由该函数下一个语句求出	MATLAB	66
trapz()	对已知数据点用梯形法求数值积分，精度不高，更好的见第 8.1.2 节	MATLAB	68
quadl()	数值积分函数，精度要求不高时还可以使用 quad() 函数	MATLAB	70
dblquad()	矩形区域的二重数值积分	MATLAB	74
quad2dggen()	非矩形区域的二重数值积分	NIT	74
triplequad()	长方体区域的三重数值积分	MATLAB	76
段落	由式 (3-5-3) 求解第一类曲线积分	自编	77
段落	由式 (3-5-4),(3-5-3) 求解第二类曲线积分	自编	78
段落	已知曲面显式表达式，由式 (3-5-7) 求解第一类曲面积分	自编	79
段落	已知曲面参数方程，由式 (3-5-9) 求解第一类曲面积分	自编	79
段落	由式 (3-5-14),(3-5-16) 求解第二类曲面积分	自编	80

- Issac Newton 和 Gattfried Wilhelm Leibnitz 创立的微积分学是很多科学科学的基础，借助 MATLAB 语言的符号运算工具箱可以直接对微积分学中最常见的问题，如单变量与多变量微积分、极限、级数求和、Taylor 幂级数展开、Fourier 级数展开等问题直接求解。
- 如果只有实验数据而未知函数原型，则需要通过数值微分的方法求其各阶微分函数。本章介绍了中心差分算法及 MATLAB 实现，经验证有很好的精度。
- 本章还给出了各种数值积分算法，介绍并比较了一般定积分、重积分的数值算法及其 MATLAB 现成函数，可以很好地用数值方法求出所需积分的解。
- 列出了两类曲线积分、两类曲面积分的公式，并通过例题演示了直接求解这些问题的方法。有了这些内容，读者可以容易地由计算机计算出这些积分问题。

3.7 习 题

1 试求出如下极限。

① $\lim\limits_{x\to\infty}(3^x+9^x)^{\frac{1}{x}}$， ② $\lim\limits_{x\to\infty}\dfrac{(x+2)^{x+2}(x+3)^{x+3}}{(x+5)^{2x+5}}$

2 试求下面的双重极限。

① $\displaystyle\lim_{\substack{x\to -1\\ y\to 2}} \frac{x^2y+xy^3}{(x+y)^3}$， ② $\displaystyle\lim_{\substack{x\to 0\\ y\to 0}} \frac{xy}{\sqrt{xy+1}-1}$， ③ $\displaystyle\lim_{\substack{x\to 0\\ y\to 0}} \frac{1-\cos\left(x^2+y^2\right)}{\left(x^2+y^2\right)\mathrm{e}^{x^2+y^2}}$

3 求出下面函数的导数。

① $y(x)=\sqrt{x\sin x\sqrt{1-\mathrm{e}^x}}$， ② $y(t)=\sqrt{\dfrac{(x-1)(x-2)}{(x-3)(x-4)}}$

③ $\operatorname{atan}\dfrac{y}{x}=\ln(x^2+y^2)$， ④ $y(x)=-\dfrac{1}{na}\ln\dfrac{x^n+a}{x^n}$, $n>0$

4 试求出 $y=\dfrac{1-\sqrt{\cos ax}}{x\left(1-\cos\sqrt{ax}\right)}$ 函数的 10 阶导数。

5 在高等数学中，求解分子和分母均同时为 0 或 ∞ 的分式极限时可使用 L'Hôpital 法则，即对分子分母分别求导数，再由比值得出。试用该法则求 $\displaystyle\lim_{x\to 0}\frac{\ln(1+x)\ln(1-x)-\ln(1-x^2)}{x^4}$，并和直接求出的极限结果相比较。

6 已知参数方程 $\begin{cases} x=\ln\cos t\\ y=\cos t-t\sin t\end{cases}$，试求出 $\dfrac{\mathrm{d}y}{\mathrm{d}x}$ 和 $\left.\dfrac{\mathrm{d}^2y}{\mathrm{d}x^2}\right|_{t=\pi/3}$。

7 假设 $u=\cos^{-1}\sqrt{\dfrac{x}{y}}$，试验证 $\dfrac{\partial^2 u}{\partial x\partial y}=\dfrac{\partial^2 u}{\partial y\partial x}$。

8 设 $\begin{cases} xu+yv=0\\ yu+xv=1\end{cases}$，试求解 $\dfrac{\partial^2 u}{\partial x\partial y}$。

9 假设 $f(x,y)=\displaystyle\int_0^{xy}\mathrm{e}^{-t^2}\mathrm{d}t$，试求 $\dfrac{x}{y}\dfrac{\partial^2 f}{\partial x^2}-2\dfrac{\partial^2 f}{\partial x\partial y}+\dfrac{\partial^2 f}{\partial y^2}$。

10 假设已知函数矩阵 $\boldsymbol{f}(x,y,z)=\begin{bmatrix}3x+\mathrm{e}^y z\\ x^3+y^2\sin z\end{bmatrix}$，试求出其 Jacobi 矩阵。

11 试求解下面的不定积分问题。

① $I(x)=-\displaystyle\int\frac{3x^2+a}{x^2\left(x^2+a\right)^2}\mathrm{d}x$， ② $I(x)=\displaystyle\int\frac{\sqrt{x(x+1)}}{\sqrt{x}+\sqrt{1+x}}\mathrm{d}x$

③ $I(x)=\displaystyle\int x\mathrm{e}^{ax}\cos bx\mathrm{d}x$， ④ $I(t)=\displaystyle\int \mathrm{e}^{ax}\sin bx\sin cx\mathrm{d}x$

12 试求出下面的定积分或无穷积分。

① $I=\displaystyle\int_0^{\infty}\frac{\cos x}{\sqrt{x}}\mathrm{d}x$， ② $I=\displaystyle\int_0^1\frac{1+x^2}{1+x^4}\mathrm{d}x$

13 假设 $f(x)=\mathrm{e}^{-5x}\sin(3x+\pi/3)$，试求出积分函数 $R(t)=\displaystyle\int_0^t f(x)f(t+x)\mathrm{d}x$。

14 对 a 的不同取值试求出 $I=\displaystyle\int_0^{\infty}\frac{\cos ax}{1+x^2}\mathrm{d}x$。

15 试对下面函数进行 Fourier 幂级数展开。

① $f(x)=(\pi-|x|)\sin x$, $-\pi\leqslant x<\pi$； ② $f(x)=\mathrm{e}^{|x|}$, $-\pi\leqslant x<\pi$；

③ $f(x)=\begin{cases}2x/l, & 0<x<l/2\\ 2(l-x)/l, & l/2<x<l\end{cases}$，且 $l=\pi$。

16 试求出下面函数的 Taylor 幂级数展开。

① $\displaystyle\int_0^x \frac{\sin t}{t}\mathrm{d}t$， ② $\ln\left(\dfrac{1+x}{1-x}\right)$， ③ $\ln\left(x+\sqrt{1+x^2}\right)$， ④ $(1+4.2x^2)^{0.2}$。

⑤ $\mathrm{e}^{-5x}\sin(3x+\pi/3)$ 分别关于 $x=0$、$x=a$ 的幂级数展开。

⑥ 对 $f(x,y)=\dfrac{1-\cos\left(x^2+y^2\right)}{\left(x^2+y^2\right)\mathrm{e}^{x^2+y^2}}$ 关于 $x=1,y=0$ 进行二维 Taylor 幂级数展开。

17 试求下面级数的前 n 项及无穷项的和。

① $\dfrac{1}{1\times 6}+\dfrac{1}{6\times 11}+\cdots+\dfrac{1}{(5n-4)(5n+1)}+\cdots$

② $\left(\dfrac{1}{2}+\dfrac{1}{3}\right)+\left(\dfrac{1}{2^2}+\dfrac{1}{3^2}\right)+\cdots+\left(\dfrac{1}{2^n}+\dfrac{1}{3^n}\right)+\cdots$

18 试求出下面的极限。

① $\displaystyle\lim_{n\to\infty}\left[\frac{1}{2^2-1}+\frac{1}{4^2-1}+\frac{1}{6^2-1}+\cdots+\frac{1}{(2n)^2-1}\right]$，

② $\displaystyle\lim_{n\to\infty} n\left(\frac{1}{n^2+\pi}+\frac{1}{n^2+2\pi}+\frac{1}{n^2+3\pi}+\cdots+\frac{1}{n^2+n\pi}\right)$。

19 试对下面数值描述的函数求取各阶数值微分，并用梯形法求取定积分。

x_i	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1	1.2
y_i	0	2.2077	3.2058	3.4435	3.241	2.8164	2.311	1.8101	1.3602	0.98172	0.67907	0.4473	0.27684

20 试求出以下的曲线积分。

① $\displaystyle\int_l (x^2+y^2)\mathrm{d}s$，$l$ 为曲线 $x=a(\cos t+t\sin t), y=a(\sin t-t\cos t), (0\leqslant t\leqslant 2\pi)$。

② $\displaystyle\int_l (yx^3+\mathrm{e}^y)\mathrm{d}x+(xy^3+x\mathrm{e}^y-2y)\mathrm{d}y$，其中，$l$ 为 $a^2x^2+b^2y^2=c^2$ 正向上半椭圆。

③ $\displaystyle\int_l y\mathrm{d}x-x\mathrm{d}y+(x^2+y^2)\mathrm{d}z$，$l$ 为曲线 $x=e^t, y=\mathrm{e}^{-t}, z=at$，$0\leqslant t\leqslant 1$，$a>0$。

④ $\displaystyle\int_l (\mathrm{e}^x\sin y-my)\mathrm{d}x+(\mathrm{e}^x\cos y-m)\mathrm{d}y$，其中，$l$ 为由 $(a,0)$ 点到 $(0,0)$ 再经 $x^2+y^2=ax$ 上正向半圆周构成的曲线。

21 试求出下面的曲面积分。

① $\displaystyle\int_S \left(2x+\frac{4y}{3}+z\right)\mathrm{d}s$，$S$ 为平面 $\dfrac{x}{2}+\dfrac{y}{3}+\dfrac{z}{4}=1$ 与 $x=-1, y=-2, z=-3$ 围成的表面。

② $\displaystyle\int_S x^2y^2z\mathrm{d}x\mathrm{d}y$，其中，$S$ 为半球面 $z=\sqrt{R^2-x^2-y^2}$ 的下侧。

第 4 章　线性代数问题的计算机求解

线性代数问题是科学技术中最常见的数学问题，很多理论和应用都是建立在线性代数的基础上的，因此解决线性代数问题是很有意义的。然而经典线性代数的课程中介绍的都是手工推导的方法，不适合于高阶矩阵的分析与计算，所以需要计算机数学语言来解决这些高阶问题。

很多计算机数学语言，如 MATLAB 语言，都起源于对线性代数问题的研究。早期的线性代数计算问题侧重于数值解法，很多数学软件包也都是从线性代数的计算开始的。例如，国际上最著名的 EISPACK 是求解矩阵特征值问题的软件包，LINPACK 是求解一般线性代数问题的软件包，目前最新的 LaPACK 也是解决线性代数计算的软件包。随着计算机科学的发展，当前能解决矩阵分析与运算问题的计算机数学语言已经不局限于数值线性代数方法了，逐渐也可以求解解析解问题。Mathmetica 和 Maple 等大型计算机数学语言都已经能直接求解线性代数的解析解问题。MATLAB 语言的符号运算工具箱可以调用 Maple 的各种解析运算功能，可以很好地解决线性代数的解析解运算问题。

本章将在第 4.1 节中将介绍矩阵的输入方法，可以用简单的函数直接输入如单位矩阵、伴随矩阵、Hankel 矩阵等特殊的矩阵，并将介绍用于解析解的符号矩阵的输入方法，为解决线性代数问题的求解打下良好的基础。第 4.2 节将介绍矩阵分析的基本概念及求解函数，例如矩阵的行列式、秩、迹、范数、逆矩阵、特征值与特征向量矩阵等，为矩阵的初步分析做准备。第 4.3 节介绍各种各样的矩阵分解方法，例如矩阵的相似变换基本概念、矩阵的正交分解、三角分解、奇异值分解等，利用矩阵分解的方法可以简化矩阵分析。第 4.4 节介绍各种各样的矩阵方程的解析解与数值解方法与结果检验方法，包括线性代数方程的可解性条件与通解求取、Lyapunov 方程、Sylvester 方程的解析解和数值解法、Riccati 方程的数值解法将在本节中给出，并给出一般 Sylvester 方程解析解的求解程序。第 4.5 节将研究矩阵元素的非线性运算及矩阵函数求解的问题，给出求解指数矩阵、三角函数矩阵以及一般矩阵函数和复合矩阵函数的解析解应用程序。从理论上说，本书提供的矩阵函数解析解程序可以用于任意复杂的矩阵函数解析运算。

4.1　特殊矩阵的输入

4.1.1　数值矩阵的输入

MATLAB 语言中固然可以通过最底层的语句逐行输入一个矩阵，但这样的方法对具有某种特殊结构的矩阵来说显得很烦琐。例如，想输入单位矩阵，再采用逐个元素输入的方式是很耗时的，故应该考虑采用 MATLAB 支持的现成函数 eye() 来输入特殊矩阵。下面将介绍一些特殊矩阵的输入方法。

4.1.1.1 零矩阵、幺矩阵及单位矩阵

在一般的矩阵理论中，把所有元素都为零的矩阵定义成零矩阵，把元素全为 1 的矩阵称为幺矩阵，把主对角线元素均为 1，而其他元素全部为 0 的方阵称为单位矩阵。这里进一步扩展单位矩阵的定义，使其为 $m \times n$ 的矩阵。零矩阵、幺矩阵和扩展单位矩阵的 MATLAB 生成函数分别为

> ***A*=zeros(*n*)，*B*=ones(*n*)，*C*=eye(*n*)**　　生成 $n \times n$ 方阵
> ***A*=zeros(*m*,*n*)；*B*=ones(*m*,*n*)；*C*=eye(*m*,*n*)**　　生成 $m \times n$ 矩阵
> ***A*=zeros(size(*B*))**　　生成和矩阵 $\boldsymbol{B}$ 同样维数的矩阵

【例 4-1】下面的语句可以生成一个 3×8 的零矩阵 $\boldsymbol{A}$，并可以生成一个和 $\boldsymbol{A}$ 维数相同的扩展单位阵 $\boldsymbol{B}$。可见，这样特殊矩阵的输入还是很容易的。

```
>> A=zeros(3,8),    % 零矩阵输入
A =
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
>> B=eye(size(A))   % 单位矩阵输入
B =
     1     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0
     0     0     1     0     0     0     0     0
```

函数 zeros() 和 ones() 还可用于多维数组的生成，例如，zeros(3,4,5) 将生成一个 3×4×5 的三维数组，其元素全部为 0。

4.1.1.2 随机元素矩阵

顾名思义，随机元素矩阵的各个元素是随机产生的。如果矩阵的随机元素满足 $[0,1]$ 区间上的均匀分布，则可以由 MATLAB 函数 rand() 来生成，该函数通常的调用格式为

> ***A*=rand(*n*)**　　生成 $n \times n$ 阶标准均匀分布伪随机数方阵
> ***A*=rand(*n*,*m*)**　　生成 $n \times m$ 阶标准均匀分布伪随机数矩阵

函数 rand() 还可以用于多维数组的生成。满足标准正态分布的随机数矩阵可以由 randn() 函数获得，当然也可以使用 rand(size(*A*)) 形式调用该函数。

这里的随机数实际上是“伪随机数”。所谓伪随机数，就是通过某种数学公式生成的、满足某些随机指标的数据。这样的随机数是可以重复的，与某些用电子方法获得的不可重复的随机数是不同的。

更一般地，如果想生成 (a,b) 区间上均匀分布的随机数，则可以先用 ***V*=rand(*n*,*m*)** 命令生成一个 $(0,1)$ 上的均匀分布随机数矩阵 $\boldsymbol{V}$，再用 $\boldsymbol{V}_1 = a + (b-a)*\boldsymbol{V}$ 语句则可以生成满足需要的矩阵 $\boldsymbol{V}_1$。如果想生成满足 $N(\mu,\sigma^2)$ 的正态分布的随机数，则可以先用 ***V*=randn(*n*,*m*)** 命令生成标准随机分布的随机数矩阵 $\boldsymbol{V}$，再用 $\boldsymbol{V}_1 = \mu + \sigma*\boldsymbol{V}$ 命令就可

以转换成所需的矩阵。

4.1.1.3 对角元素矩阵

对角矩阵是一种特殊的矩阵，这种矩阵的主对角线元素可以为 0 或非 0 元素，而非对角线元素的值均为 0。对角矩阵的数学描述方法为 $\mathrm{diag}(\alpha_1,\alpha_2,\cdots,\alpha_n)$，其中对角矩阵的矩阵表示为

$$\mathrm{diag}(\alpha_1,\alpha_2,\cdots,\alpha_n)=\begin{bmatrix}\alpha_1 & & & \\ & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_n\end{bmatrix} \tag{4-1-1}$$

MATLAB 提供了对角矩阵的生成函数 diag()。该函数的调用格式为

***A*=diag(*V*)**	已知向量生成对角矩阵
***V*=diag(*A*)**	已知矩阵提取对角元素列向量
***A*=diag(*V*,*k*)**	生成主对角线上第 k 条对角线为 $\boldsymbol{V}$ 的矩阵

【例 4-2】MATLAB 中的 diag() 函数是很有特色的，其不同方式执行不同的任务。例如，

```
>> C=[1 2 3]; V=diag(C)
V =
     1     0     0
     0     2     0
     0     0     3
>> V1=diag(V)'    % 将列向量通过转置变换成行向量
V1 =
     1     2     3
>> C=[1 2 3]; V=diag(C,2)
V =
     0     0     1     0     0
     0     0     0     2     0
     0     0     0     0     3
     0     0     0     0     0
     0     0     0     0     0
```

在实际应用中还可以取 k 为负值，表示主对角线下数的第 k 条对角线。利用这样的性质，可以容易地构造出三对角矩阵。

```
>> V=diag([1 2 3 4])+diag([2 3 4],1)+diag([5 4 3],-1)
V =
     1     2     0     0
     5     2     3     0
     0     4     3     4
     0     0     3     4
```

如果有若干个子矩阵 $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_n$，可以编写一个 `diagm()` 函数，构造块对角矩阵。该函数的清单为

```
function A=diagm(varargin)
A=[];
for i=1:length(varargin)
   A2=varargin{i};
   A(size(A,1)+1:size(A,1)+size(A2,1),size(A,2)+1:size(A,2)+size(A2,2))=A2;
end
```

该函数的调用格式为

$\boldsymbol{A}$=diagm($\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\cdots$, $\boldsymbol{A}_n$)

其中，子矩阵个数是任意多的。该函数可以得出块对角矩阵

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & & & \\ & \boldsymbol{A}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{A}_n \end{bmatrix} \tag{4-1-2}$$

4.1.1.4 Hankel 矩阵

Hankel 矩阵的一般形式如下：

$$\boldsymbol{H} = \begin{bmatrix} c_1 & c_2 & \cdots & c_m \\ c_2 & c_3 & \cdots & c_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ c_n & c_{n+1} & \cdots & c_{n+m-1} \end{bmatrix} \tag{4-1-3}$$

如果 $n \to \infty$，则可以构造无穷型 Hankel 矩阵。Hankel 矩阵是对称矩阵，且其反对角线上所有的元素都相同。

在 MATLAB 语言中，给定两个向量 $\boldsymbol{C}$ 和 $\boldsymbol{R}$，如果用

$\boldsymbol{H}$=hankel($\boldsymbol{C}$, $\boldsymbol{R}$)

来生成 $\boldsymbol{H}$，则首先将 $\boldsymbol{H}$ 矩阵的第一列的各个元素定义为 $\boldsymbol{C}$ 向量，将最后一行各个元素定义为 $\boldsymbol{R}$，这样就可以依照 Hankel 矩阵反对角线上元素相等这一特性来写出相应的 Hankel 矩阵。根据 Hankel 矩阵的性质，其最后一行的第一个元素应该等于第 1 列的最后一个元素，如果冲突将给出元素冲突的警告信息。

如果已知一个向量 $\boldsymbol{C}$，则也可以由 hankel($\boldsymbol{C}$) 函数来构造出一个 Hankel 矩阵。将 $\boldsymbol{H}$ 矩阵的第一列的各个元素定义为 $\boldsymbol{C}$ 向量，这样就可以依照 Hankel 矩阵反对角线上元素相等这一特性来写出相应的 Hankel 矩阵，使得下三角矩阵均为 0。

【例 4-3】试用 MATLAB 语句输入下面给出的 Hankel 矩阵 $\boldsymbol{H}$。

$$\boldsymbol{H} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

【求解】 分析给出的矩阵，可以用向量分别表示该矩阵的首列和最后一行，$\boldsymbol{C}$=[1,2,3]，$\boldsymbol{R}$=[3 4 5 6 7 8 9]，则可以由下面语句可以生成 Hankel 矩阵为

```
>> C=[1 2 3]; R=[3 4 5 6 7 8 9]; H=hankel(C,R)
H =
     1     2     3     4     5     6     7
     2     3     4     5     6     7     8
     3     4     5     6     7     8     9
```

如果只给出一个向量 $\boldsymbol{C}$=[1 2 3]，则可以生成下三角矩阵为 0 的方阵

```
>> C=[1 2 3]; H1=hankel(C)
H1 =
     1     2     3
     2     3     0
     3     0     0
```

4.1.1.5 Hilbert 矩阵及逆 Hilbert 矩阵

Hilbert 矩阵是一类特殊矩阵，它的第 (i,j) 元素的值满足 $h_{i,j}=1/(i+j-1)$，这时一个 $n\times n$ 阶的 Hilbert 矩阵可以写成

$$\boldsymbol{H}=\begin{bmatrix}1 & 1/2 & 1/3 & \cdots & 1/n\\ 1/2 & 1/3 & 1/4 & \cdots & 1/(n+1)\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1/n & 1/(n+1) & 1/(n+2) & \cdots & 1/(2n-1)\end{bmatrix} \tag{4-1-4}$$

产生 Hilbert 矩阵的 MATLAB 函数为

$\boldsymbol{A}$=hilb(n)

其中，n 为要产生的矩阵阶次。

高阶 Hilbert 矩阵一般为坏条件的矩阵，所以直接对之求逆一般往往会引出浮点溢出的现象。MATLAB 提供了直接求取逆 Hilbert 矩阵的算法及函数，其调用格式为

$\boldsymbol{B}$=invhilb(n)

由于 Hilbert 矩阵本身接近奇异的性质，所以在处理该矩阵时建议尽量采用符号运算工具箱，而采用数值解时应该检验结果的正确性。

4.1.1.6 Vandermonde 矩阵

假设有一个序列 $\boldsymbol{C}$，其各个元素满足 $\{c_1,c_2,\cdots,c_n\}$，则可以写出一个矩阵，其第 (i,j) 元素满足 $v_{i,j}=c_i^{n-j}$，$i,j=1,2,\cdots,n$。这样可以构成一个矩阵

$$\boldsymbol{V}=\begin{bmatrix}c_1^{n-1} & c_1^{n-2} & \cdots & c_1 & 1\\ c_2^{n-1} & c_2^{n-2} & \cdots & c_2 & 1\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ c_n^{n-1} & c_n^{n-2} & \cdots & c_n & 1\end{bmatrix} \tag{4-1-5}$$

该矩阵称作 Vandermonde 矩阵。如果已知一个向量 $\boldsymbol{C}$，则可以由 MATLAB 提供的 vander() 函数来构造一个 Vandermonde 矩阵。该函数的调用格式为

```
V=vander(C)
```

【例 4-4】若向量 $\boldsymbol{C}$=[1,2,3,4,5]，则可以得出该向量对应的 Vandermonde 矩阵为

```
>> C=[1, 2, 3, 4, 5];  V=vander(C)
V =
     1     1     1     1     1
    16     8     4     2     1
    81    27     9     3     1
   256    64    16     4     1
   625   125    25     5     1
```

4.1.1.7 **伴随矩阵**

假设有一个首一化的多项式

$$P(s)=s^n+a_1s^{n-1}+a_2s^{n-2}+\cdots+a_{n-1}s+a_n \tag{4-1-6}$$

则可以写出一个伴随矩阵

$$\boldsymbol{A}_c=\begin{bmatrix}-a_1 & -a_2 & \cdots & -a_{n-1} & -a_n\\ 1 & 0 & \cdots & 0 & 0\\ 0 & 1 & \cdots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & \cdots & 1 & 0\end{bmatrix} \tag{4-1-7}$$

生成伴随矩阵的 MATLAB 函数调用格式为

```
B=compan(p)
```

其中，$\boldsymbol{p}$ 为一个多项式向量，该函数将自动对多项式进行首一化处理。

【例 4-5】考虑一个多项式 $P(s)=2s^4+4s^2+5s+6$，试写出该多项式的伴随矩阵。

【求解】先输入特征多项式，则伴随矩阵可以通过下面的语句建立起来，赋给 $\boldsymbol{A}$ 矩阵。

```
>> P=[2 0 4 5 6]; A=compan(P)
A =
         0   -2.0000   -2.5000   -3.0000
    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
```

4.1.2 符号矩阵的输入

如果已经建立起了数值矩阵 $\boldsymbol{A}$，则可以由

```
B=sym(A)
```

语句将其转换成符号矩阵。这样，所有数值矩阵均可以通过这样的形式转换成符号矩阵，可以利用符号运算工具箱获得更高精度的解。

对于一些特殊矩阵形式，如 Vandermonde 矩阵、Hankel 矩阵及伴随矩阵，符号运算工具箱不直接支持它们，所以需要编写下面的一些函数，将其置于 @sym 目录下，这样才可以对符号向量建立起这些符号矩阵。

参考 MATLAB 语言对数字矩阵生成的相应函数，可以改写出适合符号运算的新函数。例如，可以编写出生成伴随矩阵的 MATLAB 函数 compan()。

```
function A=compan(c)
c=c(:).'; A=sym(diag(ones(1,n-2),-1));
A(1,:)=-c(2:n)./c(1);
```

【例 4-6】试用解析方法建立起下面多项式的伴随矩阵。

$$P(\lambda)=a_1\lambda^9+a_2\lambda^8+a_3\lambda^7+\cdots+a_8\lambda^2+a_9\lambda+a_{10}$$

【求解】由上面编写的函数，可以先申明符号变量，并以向量形式输入多项式，最后可以通过下面的语句直接建立所需的伴随矩阵。

```
>> syms a1 a2 a3 a4 a5 a6 a7 a8 a9 a10
   A=compan([a1 a2 a3 a4 a5 a6 a7 a8 a9 a10])
A =
   [ -a2/a1, -a3/a1, -a4/a1, -a5/a1, -a6/a1, -a7/a1, -a8/a1, -a9/a1, -a10/a1]
   [      1,      0,      0,      0,      0,      0,      0,      0,       0]
   [      0,      1,      0,      0,      0,      0,      0,      0,       0]
   [      0,      0,      1,      0,      0,      0,      0,      0,       0]
   [      0,      0,      0,      1,      0,      0,      0,      0,       0]
   [      0,      0,      0,      0,      1,      0,      0,      0,       0]
   [      0,      0,      0,      0,      0,      1,      0,      0,       0]
   [      0,      0,      0,      0,      0,      0,      1,      0,       0]
   [      0,      0,      0,      0,      0,      0,      0,      1,       0]
```

类似地，Hankel 矩阵的 MATLAB 函数 hankel() 为

```
function H=hankel(c,r)
c=c(:); nc=length(c);
if nargin==1, r=zeros(size(c)); end
r=r(:); nr = length(r);
x=[c; r((2:nr)')]; cidx=(1:nc)';
ridx=0:(nr-1); H1=cidx(:,ones(nr,1))+ridx(ones(nc,1),:);
H=x(H1);
```

还可以建立起生成 Vandermonde 矩阵的 MATLAB 函数 vander()，其格式与 MATLAB 语言中的数值函数完全一致。

```
function A=vander(v)
n=length(v); v=v(:); A=sym(ones(n));
for j=n-1:-1:1, A(:,j)=v.*A(:,j+1); end
```

4.2 矩阵基本分析

4.2.1 矩阵基本概念与性质

4.2.1.1 行列式

矩阵 $\boldsymbol{A}=\{a_{ij}\}$ 的行列式定义为

$$\boldsymbol{D}=|\boldsymbol{A}|=\det(\boldsymbol{A})=\sum(-1)^k a_{1k_1}a_{2k_2}\cdots a_{nk_n} \tag{4-2-1}$$

式中，$k_1,k_2,\cdots,k_n$ 是将序列 $1,2,\cdots,n$ 的元素交换 k 次所得出的一个序列，每个这样的序列称为一个置换 (permutation)；而 Σ 表示对 $k_1,k_2,\cdots,k_n$ 取遍 $1,2,\cdots,n$ 的所有排列的求和。

计算矩阵的行列式有多种算法，在 MATLAB 中采用的方法是对原矩阵 $\boldsymbol{A}$ 进行三角分解(又称为 LU 分解，后面将介绍)，将其分解成一个上三角矩阵 $\boldsymbol{U}$ 和一个下三角矩阵 $\boldsymbol{L}$ 的积，即 $\boldsymbol{A}=\boldsymbol{LU}$，这样可以先求出 $\boldsymbol{L}$ 矩阵的行列式。注意，在这一矩阵中只有一种非 0 的排列方式且其行列式的值 s 为 1 或 -1。同样，因为 $\boldsymbol{U}$ 为上三角矩阵，所以其行列式的值为该矩阵主对角线元素之积，即 $\boldsymbol{A}$ 矩阵行列式为 $\det(\boldsymbol{A})=s\prod_{i=1}^n u_{ii}$。MATLAB 提供了内在函数 det()，其调用格式很直观，为

d=det($\boldsymbol{A}$)

利用它可以直接求取矩阵 $\boldsymbol{A}$ 的行列式。该函数同样适用于符号矩阵 $\boldsymbol{A}$。

【例 4-7】试求出下面矩阵的行列式。

$$\boldsymbol{A}=\begin{bmatrix}16 & 2 & 3 & 13\\ 5 & 11 & 10 & 8\\ 9 & 7 & 6 & 12\\ 4 & 14 & 15 & 1\end{bmatrix}$$

【求解】由下面的语句可以立即得出矩阵的行列式为 0。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; det(A)
ans =
     0
```

【例 4-8】从第 1 章给出的例子可知，高阶 Hilbert 矩阵是接近奇异的矩阵。试用解析解方法计算出 20×20 的 Hilbert 矩阵的行列式。

【求解】首先用 hilb() 函数可以定义一个 20×20 的 Hilbert 矩阵，将其转换成符号矩阵，则 MATLAB 的 det() 函数会自动采用解析解法求出其行列式的值。

```
>> tic, A=sym(hilb(20)); det(A), toc
ans =
```

```
1/237745471676853450909164424342761644017541983775348649303318533123441975931
  064458518758576681657377344056575986726555897176563841971079330338658232414
  981124102355448916615471780963525779783680000000000000000000000000000000000
Elapsed time is 0.862000 seconds.
```

亦即可以得出如下的行列式的解析解及近似值为

$$\det(\boldsymbol{H})=\frac{1}{\underbrace{2377454716\cdots3680000000000000000000000000000000000}_{225\text{ 位，因排版限制省略了中间的数字}}}\simeq 4.206179\times 10^{-224}$$

从计算结果还可以看出，利用解析方法在 1 秒钟内就可以得出原问题的解析解，因为这里采用的方法是三角矩阵分解的方法，而不是代数余子式算法。

4.2.1.2　矩阵的迹

假设一个方阵为 $\boldsymbol{A}=\{a_{ij}\}, i,j=1,2,\cdots,n$，则矩阵 $\boldsymbol{A}$ 的迹定义为

$$\mathrm{tr}(\boldsymbol{A})=\sum_{i=1}^{n}a_{ii} \tag{4-2-2}$$

亦即矩阵的迹为该矩阵对角线上各个元素之和。由代数理论可知，矩阵的迹和该矩阵的特征值之和是相同的，矩阵 $\boldsymbol{A}$ 的迹可以由 MATLAB 函数 `trace()` 求出，该函数的调用和数学表示相似，如下：

```
t=trace(A)
```

例 4-7 中矩阵的迹可以由 MATLAB 语句直接求出：trace($\boldsymbol{A}$)=34。

4.2.1.3　矩阵的秩

若矩阵所有的列向量中共有 r_c 个线性无关，则称矩阵的列秩为 r_c。如果 $r_c=m$，则称 $\boldsymbol{A}$ 为列满秩矩阵。相应地，若矩阵 $\boldsymbol{A}$ 的行向量中有 r_r 个是线性无关的，则称矩阵 $\boldsymbol{A}$ 的行秩为 r_r。如果 $r_r=n$，则称 $\boldsymbol{A}$ 为行满秩矩阵。可以证明，矩阵的行秩和列秩是相等的，故称之为矩阵的秩，记作

$$\mathrm{rank}(\boldsymbol{A})=r_c=r_r \tag{4-2-3}$$

这时，矩阵的秩为 rank($\boldsymbol{A}$)。矩阵的秩也表示该矩阵中行列式不等于 0 的子式的最大阶次。所谓子式，即为从原矩阵中任取 k 行及 k 列所构成的子矩阵。

矩阵求秩的算法也是多种多样的，其区别是有的算法是稳定的，而有的算法可能因矩阵的条件数变化不是很稳定。MATLAB 中采用的算法是基于矩阵的奇异值分解的算法[7]。首先对矩阵进行奇异值分解，得出矩阵 $\boldsymbol{A}$ 的 n 个奇异值 $\sigma_i, i=1,2,\cdots,n$，在这 n 个奇异值中找出大于给定误差限 ε 的个数 r，这时 r 就可以认为是 $\boldsymbol{A}$ 矩阵的秩。

MATLAB 提供了一个内在函数 `rank()`，用它可以求取给定矩阵的秩。该函数的调用格式为

```
r=rank(A)     % 用默认的精度求数值秩
r=rank(A,ε)   % 给定精度 ε 下求数值秩
```

其中，$\boldsymbol{A}$ 为给定矩阵，ε 为机器精度。符号运算工具箱中也提供了 rank() 函数，可以求出数值矩阵秩的解析解，其调用格式由前面的方法完全一致。

【例 4-9】试求出例 4-7 中给出的 $\boldsymbol{A}$ 矩阵的秩。

【求解】用 rank($\boldsymbol{A}$) 函数可以得出该矩阵的秩为

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];  rank(A)
ans =
     3
```

该矩阵的秩为 3，小于矩阵的阶次，故矩阵 $\boldsymbol{A}$ 是非满秩矩阵。

【例 4-10】现在考虑例 4-8 中给出的 20×20 Hilbert 矩阵，考虑用数值方法和解析方法分别求该矩阵的秩，并比较其正确性。

【求解】先考虑数值方法，应该给出命令

```
>> H=hilb(20); rank(H)
ans =
     13
```

故而可以得出结论。因为该矩阵的秩和矩阵阶次相差太多，所以 $\boldsymbol{H}$ 矩阵为非满秩矩阵。其实该函数对一些接近奇异的矩阵可能出现错误结论，用数值解的方法应该注意。如果有可能应该采用解析解的方法求解该问题。

```
>> H=sym(hilb(20)); rank(H)  % 可见原矩阵为非奇异矩阵
H =
    20
```

4.2.1.4 矩阵范数

矩阵的范数是对矩阵的一种测度。在介绍矩阵的范数之前，首先要介绍向量范数的基本概念。如果对线性空间中的一个向量 $\boldsymbol{x}$ 存在一个函数 $\rho(\boldsymbol{x})$ 满足下面 3 个条件：

① $\rho(\boldsymbol{x})\geqslant 0$ 且 $\rho(\boldsymbol{x})=0$ 的充要条件是 $\boldsymbol{x}=\boldsymbol{0}$

② $\rho(a\boldsymbol{x})=|a|\rho(\boldsymbol{x})$, a 为任意标量

③ 对向量 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 有 $\rho(\boldsymbol{x}+\boldsymbol{y})\leqslant\rho(\boldsymbol{x})+\rho(\boldsymbol{y})$

则称 $\rho(\boldsymbol{x})$ 为 $\boldsymbol{x}$ 向量的范数。范数的形式是多种多样的。可以证明，下面给出的一族式子都满足上述的 3 个条件。

$$||\boldsymbol{x}||_p=\left(\sum_{i=1}^{n}|x_i|^p\right)^{1/p},\ p=1,2,\cdots,\text{且}\ ||\boldsymbol{x}||_\infty=\max_{1\leqslant i\leqslant n}|x_i| \tag{4-2-4}$$

这里用到了向量范数的记号 $||\boldsymbol{x}||_p$。

矩阵的范数定义比向量的稍复杂一些，其数学定义为：对于任意的非零向量 $\boldsymbol{x}$，矩阵 $\boldsymbol{A}$ 的范数为

$$||\boldsymbol{A}||=\sup_{\boldsymbol{x}\neq 0}\frac{||\boldsymbol{A}\boldsymbol{x}||}{||\boldsymbol{x}||} \tag{4-2-5}$$

和向量的范数一样，对矩阵来说也有常用的范数定义方法

$$||\boldsymbol{A}||_1 = \max_{1\leqslant j\leqslant n}\sum_{i=1}^{n} | a_{ij} |, \ ||\boldsymbol{A}||_2 = \sqrt{s_{\max}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})}, \ ||\boldsymbol{A}||_\infty = \max_{1\leqslant i\leqslant n}\sum_{j=1}^{n} | a_{ij} | \tag{4-2-6}$$

其中，$s(\boldsymbol{X})$ 为 $\boldsymbol{X}$ 矩阵的特征值，而 $s_{\max}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})$ 为 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 矩阵的最大特征值。事实上，$||\boldsymbol{A}||_2$ 还等于 $\boldsymbol{A}$ 矩阵的最大奇异值。

MATLAB 提供了求取矩阵范数的函数 `norm()`，允许求各种意义下的矩阵范数。该函数的调用格式为

N=**norm(*A*)**	求解默认的 $\|\|\boldsymbol{A}\|\|_2$
N=**norm(*A*，选项)**	选项可以为 1,2 等，具体见表 4-1

表 4-1 矩阵范数函数的选项表

选 项	意义及算法
无	矩阵的最大奇异值，即 $\|\|\boldsymbol{A}\|\|_2$
2	与默认调用方式相同，即 $\|\|\boldsymbol{A}\|\|_2$
1	矩阵的 1-范数，即 $\|\|\boldsymbol{A}\|\|_1$
`Inf` 或 `'inf'`	矩阵的无穷范数，即 $\|\|\boldsymbol{A}\|\|_\infty$
`'fro'`	矩阵的 Frobinius 范数，即 $\|\|\boldsymbol{A}\|\|_F = \sqrt{\sum(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})_{ii}}$
数值 P	对向量可取任何整数，而对矩阵只可取 1, 2, `inf` 或 `'fro'`
`-inf`	只可用于向量，$\|\|\boldsymbol{A}\|\|_{-\infty} = \min(\|\sum a_i\|)$

这样，例 4-7 中矩阵 $\boldsymbol{A}$ 的各种范数可以由下面的 MATLAB 函数直接求出。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];
   [norm(A), norm(A,2), norm(A,1), norm(A,Inf), norm(A,'fro')]
ans =
   34.0000   34.0000   34.0000   34.0000   38.6782
```

这里有两点值得注意，首先 **norm(*A*)** 和 **norm(*A*,2)** 应该给出同样的结果，因为它们都表示 $||\boldsymbol{A}||_2$，其次因为巧合，在这个例子中，$||\boldsymbol{A}||_1 = ||\boldsymbol{A}||_\infty$。但一般情况下，$||\boldsymbol{A}||_1 = ||\boldsymbol{A}||_\infty$ 不一定能满足。

遗憾的是，符号运算工具箱中未提供 `norm()` 函数。故若需要求解数值矩阵的范数，应该先将矩阵用 `double()` 函数转换成双精度数值矩阵，然后再调用数值矩阵的 `norm()` 函数。

4.2.1.5 特征多项式

引入算子 s，并构造一个矩阵 $s\boldsymbol{I}-\boldsymbol{A}$，再求出该矩阵的行列式，则可以得出一个关于算子 s 的多项式

$$C(s) = \det(s\boldsymbol{I}-\boldsymbol{A}) = s^n + c_1 s^{n-1} + \cdots + c_{n-1}s + c_n \tag{4-2-7}$$

这样的多项式 $C(s)$ 称为矩阵 $\boldsymbol{A}$ 的特征多项式。其中，系数 $c_i, i=1,2,\cdots,n$ 称为矩阵的特征多项式系数。

MATLAB 提供了求取矩阵特征多项式系数的函数 poly()，该函数的调用格式为

$\boldsymbol{C}$=poly($\boldsymbol{A}$)

返回的 $\boldsymbol{C}$ 为一个行向量，其各个分量为矩阵 $\boldsymbol{A}$ 的降幂排列的特征多项式系数。该函数的另外一种调用格式是，如果给定的 $\boldsymbol{A}$ 为向量，则假定该向量是一个矩阵的特征根，由此求出该矩阵的特征多项式系数，如果向量 $\boldsymbol{A}$ 中有无穷大或 NaN 值，则首先剔除它，再计算特征多项式的系数。

值得指出的是，如果 $\boldsymbol{A}$ 为符号矩阵，该函数仍然适用，但得出的不是系数向量，而是多项式的数学表达式本身。

【例 4-11】试求出例 4-7 中给出的 $\boldsymbol{A}$ 矩阵的特征多项式。

【求解】可以通过下面的 poly() 函数直接求出该矩阵的特征多项式。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];
   poly(A)
ans =
   1.0e+003 *
     0.0010   -0.0340   -0.0800    2.7200   -0.0000
```

用符号运算工具箱中的 poly() 函数同样可以求出矩阵的特征多项式。

```
>> A=sym(A); poly(A)
ans =
   x^4-34*x^3-80*x^2+2720*x
```

在实际应用中还有其他简单的数值方法可以精确地求出矩阵的特征多项式系数。例如，下面给出的 Fadeev-Fadeeva 递推算法也可以求出矩阵的特征多项式。

$$\begin{cases} c_k = -\dfrac{1}{k}\mathrm{tr}(\boldsymbol{A}\boldsymbol{R}_k), & k=1,2,\cdots,n \\ \boldsymbol{R}_1=\boldsymbol{I},\ \boldsymbol{R}_k=\boldsymbol{A}\boldsymbol{R}_{k-1}+c_{k-1}\boldsymbol{I}, & k=2,\cdots,n \end{cases} \tag{4-2-8}$$

该算法首先给出一个单位阵 $\boldsymbol{I}$，并将之赋给 $\boldsymbol{R}_1$，然后对每个 k 的值分别求出特征多项式参数，并更新 $\boldsymbol{R}_k$ 矩阵，最终得出矩阵的特征多项式系数 c_k。该算法可以直接由下面的 MATLAB 语句编写一个 poly1() 函数实现。

```
function c=poly1(A)
[nr,nc]=size(A);
if nc==nr % 给出若为方阵，则用 Fadeev-Fadeeva 算法求特征多项式
   I=eye(nc);  R=I; c=[1 zeros(1,nc)];
   for k=1:nc, c(k+1)=-1/k*trace(A*R); R=A*R+c(k+1)*I;
   end
elseif (nr==1 | nc==1) % 给出为向量时，构造矩阵
   A=A(isfinite(A)); n=length(A); % 除去非数或无界的特征根
```

```
    c = [1 zeros(1,n)];
    for j=1:n
       c(2:(j+1))=c(2:(j+1))-A(j).*c(1:j);
    end
 else   % 参数有误则给出错误信息
    error('Argument must be a vector or a square matrix.')
 end
```

调用新的 poly1() 函数，则可以得出如下的精确结果。

```
>> poly1(A)
ans =
        1          -34          -80         2720           0
```

【例 4-12】试推导出向量 $\boldsymbol{B}=[a_1,a_2,a_3,a_4,a_5]$ 对应的 Vandermonde 矩阵的特征多项式。

【求解】可以用 $\boldsymbol{A}$=vander($\boldsymbol{B}$) 函数构造一个 Vandermonde 矩阵 $\boldsymbol{A}$，这样就能用 poly($\boldsymbol{A}$) 函数获得该矩阵的特征多项式。

```
>> syms a1 a2 a3 a4 a5 x; A=vander([a1 a2 a3 a4 a5]);
   collect(poly(A),x) % 按 x 合并同类项，化简多项式
```

该矩阵的特征多项式数学表示为

$$\begin{aligned}\det(\boldsymbol{A}) &= x^5+(-a_3-a_1-a_5)x^4+(a_5a_1+a_3a_1+a_5a_3-2a_4^2-2a_5^2-a_2^2-a_3^2)x^3+\\&(a_4^2a_1+a_5^2a_3+a_4^2a_5+a_2^2a_5+a_4^2a_3-a_3a_1a_5-2a_2a_5a_4+a_5^2a_1-2a_2a_4a_3+2a_5^3+a_3^3)x^2+\\&(-3a_4^2a_3a_5+a_5^4+a_4^2a_5^2+a_4^4-a_5^3a_1+a_5^2a_3^2+2a_2a_5^2a_4-a_5^3a+3)x-a_5^5\end{aligned}$$

4.2.1.6 矩阵多项式的求解

矩阵多项式的数学形式为

$$\boldsymbol{B}=a_1\boldsymbol{A}^n+a_2\boldsymbol{A}^{n-1}+\cdots+a_n\boldsymbol{A}+a_{n+1}\boldsymbol{I} \tag{4-2-9}$$

其中，$\boldsymbol{A}$ 为一个给定矩阵，$\boldsymbol{I}$ 为和 $\boldsymbol{A}$ 同阶次的单位矩阵，这时返回的矩阵 $\boldsymbol{B}$ 为矩阵多项式的值。矩阵多项式的值在 MATLAB 语言环境中可以由 polyvalm() 函数求出，该函数的调用格式为

$\boldsymbol{B}$=polyvalm($\boldsymbol{a}$,$\boldsymbol{A}$)

其中，$\boldsymbol{a}$ 为多项式系数降幂排列构成的向量，即 $\boldsymbol{a}=[a_1,a_2,\cdots,a_n,a_{n+1}]$。

相应地，还可以按点运算的方式定义一种多项式运算为

$$\boldsymbol{C}=a_1\boldsymbol{x}\,.\hat{}\,n+a_2\boldsymbol{x}\,.\hat{}\,(n-1)+\cdots+a_{n+1} \tag{4-2-10}$$

这时，矩阵 $\boldsymbol{C}$ 可以由下面的语句直接计算出来。

$\boldsymbol{C}$=polyval($\boldsymbol{a}$,$\boldsymbol{x}$)

若由 MATLAB 的符号运算工具箱给出多项式 p，则可以调用 subs() 函数求出点运算意义下的多项式的值。该函数在此问题上的具体调用格式为

```
C=subs(p,s,x)
```

【例 4-13】Cayley-Hamilton 定理是矩阵理论中的一个比较重要的定理，其内容为：若矩阵 $\boldsymbol{A}$ 的特征多项式为

$$f(s)=\det(s\boldsymbol{I}-\boldsymbol{A})=a_1s^n+a_2s^{n-1}+\cdots+a_ns+a_{n+1} \tag{4-2-11}$$

则有 $f(\boldsymbol{A})=\boldsymbol{0}$，亦即

$$a_1\boldsymbol{A}^n+a_2\boldsymbol{A}^{n-1}+\cdots+a_n\boldsymbol{A}+a_{n+1}\boldsymbol{I}=\boldsymbol{0} \tag{4-2-12}$$

假设矩阵 $\boldsymbol{A}$ 为 Vandermonde 矩阵，试验证其满足 Cayley-Hamilton 定理。

【求解】可以由下面的 MATLAB 语句来验证 Cayley-Hamilton 定理。

```
>> A=vander([1 2 3 4 5 6 7])
A =
         1       1       1       1       1       1       1
        64      32      16       8       4       2       1
       729     243      81      27       9       3       1
      4096    1024     256      64      16       4       1
     15625    3125     625     125      25       5       1
     46656    7776    1296     216      36       6       1
    117649   16807    2401     343      49       7       1
>> aa=poly(A); B=polyvalm(aa, A);  norm(B)
ans =
   2.188659936727728e+006
```

由于使用的 poly() 函数会产生一定的误差，而该误差在矩阵多项式求解中导致了巨大的误差，从而得出错误结论。由此看来，poly() 函数的误差有时是不可忽略的。如果把上面语句中的 poly() 函数用前面编写的 poly1() 函数代替，则可以得出如下结果：

```
>> aa1=poly1(A); B1=polyvalm(aa1, A);  norm(B1)
ans =
     0
```

可见，由此得出的 $\boldsymbol{B}$ 矩阵就会完全等于 $\boldsymbol{0}$，故该矩阵满足 Cayley-Hamilton 定理。

4.2.1.7 符号多项式与数值多项式的转换

若已知数值多项式系数构成的向量 $\boldsymbol{P}$=[$a_1,a_2,\cdots,a_{n+1}$]，则可以通过符号运算工具箱提供的 poly2sym() 函数转换成多项式表示。若已知多项式的符号表达式，则可以由 sym2poly() 函数转换成系数向量形式。这两个函数的调用格式都是很简单的：

```
f=poly2sym(P) 或 f=poly2sym(P,x)
P=sym2poly(f)
```

【例 4-14】已知多项式 $f=s^5+2s^4+3s^3+4s^2+5s+6$，试用不同形式表示该多项式。

【求解】 该多项式可以用两种形式先定义出来。例如，可以用数值形式先定义之，则可以用相应的方式将其转换成符号型的多项式。

```
>> P=[1 2 3 4 5 6]; % 先由系数按降幂顺序排列表示多项式
   f=poly2sym(P,'v')   % 以 v 为算子表示多项式
f =
   v^5+2*v^4+3*v^3+4*v^2+5*v+6
```

显然，另一种方法是需要先表示符号形式的多项式，然后用转换函数将其转换成数值形式。

```
>> P=sym2poly(f)   % 转换成数值形式的多项式
P =
     1     2     3     4     5     6
```

4.2.2 逆矩阵与广义逆矩阵

4.2.2.1 矩阵的逆矩阵

对一个已知的 $n \times n$ 非奇异方阵 $\boldsymbol{A}$ 来说，如果有一个同样大小的 $\boldsymbol{C}$ 矩阵满足

$$\boldsymbol{AC}=\boldsymbol{CA}=\boldsymbol{I} \tag{4-2-13}$$

式中 $\boldsymbol{I}$ 为单位阵，则称 $\boldsymbol{C}$ 矩阵为 $\boldsymbol{A}$ 矩阵的逆矩阵，并记作 $\boldsymbol{C}=\boldsymbol{A}^{-1}$。

MATLAB 语言中提供了 inv() 函数，可以直接用来求取矩阵的逆矩阵，亦即

$\boldsymbol{C}$=inv($\boldsymbol{A}$)

该函数同样适用于符号变量构成的矩阵的求逆。

【例 4-15】 试求取 Hilbert 矩阵的逆矩阵。

【求解】 考虑 4 × 4 Hilbert 矩阵，调用 MATLAB 的矩阵求逆函数 inv()，则可以立即得出该矩阵的逆矩阵来。

```
>> format long; H=hilb(4); H1=inv(H)
ans =
  1.0e+003 *
   0.01600000000000  -0.11999999999999   0.23999999999998  -0.13999999999999
  -0.11999999999999   1.19999999999990  -2.69999999999976   1.67999999999984
   0.23999999999998  -2.69999999999976   6.47999999999940  -4.19999999999961
  -0.13999999999999   1.67999999999984  -4.19999999999961   2.79999999999974
```

这里还可以应用 MATLAB 的语句来检验得出的逆矩阵是否符合条件。例如，计算原矩阵 $\boldsymbol{H}$ 和求出的逆矩阵 $\boldsymbol{H}_1$ 之积，则

```
>> H*H1
ans =
   1.00000000000001   0.00000000000023  -0.00000000000045   0.00000000000023
   0.00000000000001   1.00000000000011  -0.00000000000011   0.00000000000011
   0.00000000000001                  0   1.00000000000011                  0
```

```
0.00000000000000   0.00000000000011  -0.00000000000011   1.00000000000011
```

二者之积应该为单位阵，但实际上这样的积有点误差。对大规模矩阵来说，如果用上述的检验方法显得太麻烦，所以可以将结果减去一个单位阵，对其误差用求范数的方法，对得出的矩阵进行检验，如果误差矩阵的范数是一个微小的数，则可以接受得出的逆矩阵，否则应该认为其不正确。对这个例子来说，可以计算出误差的范数为

```
>> norm(H*inv(H)-eye(size(H)))
ans =
    6.235798190375727e-013
```

从结果看，此误差虽然未小于 MATLAB 矩阵运算的一般误差 ($10^{-15} \sim 10^{-16}$ 级)，但还是比较小的，因此可以接受得出的逆矩阵。

考虑到高阶 Hilbert 矩阵接近于奇异矩阵，一般不建议用 inv() 函数直接求解，可以采用 invhilb() 函数直接产生逆矩阵。

```
>> H2=invhilb(4); norm(H*H2-eye(size(H)))
ans =
    5.684341886080802e-014
```

可见，对于低阶矩阵，用 invhilb() 计算出来的逆矩阵的精度也显著改善了。现在考虑 10×10 的 Hilbert 矩阵，则

```
>> H=hilb(10); H1=inv(H); norm(H*H1-eye(size(H)))
ans =
   0.00264500826202
>> H2=invhilb(10); norm(H*H2-eye(size(H)))
ans =
    1.612897415528547e-005
```

这样虽然得出的逆矩阵精度远高于直接求逆的精度，但还是难以达到较高的要求。进一步扩大矩阵的阶次，例如需要研究 13×13 的 Hilbert 矩阵，则

```
>> H=hilb(13); H1=inv(H); norm(H*H1-eye(size(H)))
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 2.339949e-018.
ans =
  53.23696008570294
>> H2=invhilb(13); norm(H*H2-eye(size(H)))
ans =
  11.37062973181391
```

符号运算工具箱中也对符号矩阵定义了 inv() 函数，即使对更高阶的非奇异矩阵也可以精确求解出矩阵的逆矩阵来。现在先显示 7×7 的 Hilbert 逆矩阵为

```
>> H=sym(hilb(7)); inv(H)
ans =
[      49,     -1176,        8820,      -29400,        48510,       -38808,       12012]
```

```
[  -1176,    37632,   -317520,   1128960,   -1940400,   1596672,   -504504]
[   8820,  -317520,   2857680, -10584000,   18711000, -15717240,   5045040]
[ -29400,  1128960, -10584000,  40320000,  -72765000,  62092800, -20180160]
[  48510, -1940400,  18711000, -72765000,  133402500,-115259760,  37837800]
[ -38808,  1596672, -15717240,  62092800, -115259760, 100590336, -33297264]
[  12012,  -504504,   5045040, -20180160,   37837800, -33297264,  11099088]
```

其实，用符号运算工具箱可以求解出更高阶 Hilbert 矩阵的逆矩阵。例如，求解 30 阶矩阵，可以使用下面的命令，得出精确的结果。

```
>> H=sym(hilb(30)); norm(double(H*inv(H)-eye(size(H))))
ans =
     0
```

【例 4-16】 试对例 4-7 中给出的奇异矩阵 $\boldsymbol{A}$ 求逆，并观察用数值方法对真正奇异的矩阵求逆会发生什么现象。

【求解】 首先输入该矩阵，则可以用 inv() 函数对其求逆。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];
   format long; B = inv(A)
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.306145e-017.
B =
  1.0e+014 *
   0.93824992236885   2.81474976710656  -2.81474976710656  -0.93824992236885
   2.81474976710656   8.44424930131968  -8.44424930131968  -2.81474976710656
  -2.81474976710656  -8.44424930131968   8.44424930131968   2.81474976710656
  -0.93824992236885  -2.81474976710656   2.81474976710656   0.93824992236885
```

可见，这时给出一个警告信息，提示用户该矩阵接近于奇异 (事实上，$\boldsymbol{A}$ 矩阵就是一个奇异矩阵，但通过数值算法后得出的是接近奇异的结论)，并提示得出的逆矩阵可能是不正确的，最后还列出了逆矩阵的结果。如果对上面的结果进行验算，会发现误差很大，所以逆矩阵是错误的。

```
>> norm(A*B-eye(size(A)))
ans =
   1.64081513306419
```

事实上，奇异矩阵根本不存在一个相应的逆矩阵，能满足式 (4-2-13) 中的条件。对这里给出的问题还可以试用符号运算工具箱中的函数，但由于矩阵奇异，故 inv() 函数也无能为力。

```
>> A=sym(A);  inv(A)
??? Error using ==> sym/inv
Error, (in inverse) singular matrix
```

【例 4-17】 MATLAB 的矩阵求逆函数同样适用于含有变量的矩阵。例如，对于下面的 Hankel 矩阵，可以轻易用 inv() 函数得出其逆矩阵。

```
>> syms a1 a2 a3 a4; H=hankel([a1 a2 a3 a4]); inv(H)
```

```
ans =
[     0,           0,                  0,                         1/a4]
[     0,           0,               1/a4,                 -1/a4^2*a3]
[     0,        1/a4,          -1/a4^2*a3,          -1/a4^3*(a2*a4-a3^2)]
[ 1/a4, -1/a4^2*a3, -1/a4^3*(a2*a4-a3^2), -(a1*a4^2-2*a2*a3*a4+a3^3)/a4^4]
```

4.2.2.2 矩阵的广义逆

前面已经介绍过，即使用解析解求解的符号运算工具箱对奇异矩阵的求逆也是无能为力的，因为其逆矩阵根本不存在。另外，长方形的矩阵有时也会涉及到求逆的问题，这样就需要定义一种新的“逆矩阵”。对于要研究的矩阵 $\boldsymbol{A}$，如果存在一个矩阵 $\boldsymbol{N}$，它满足

$$\boldsymbol{ANA}=\boldsymbol{A} \tag{4-2-14}$$

则 $\boldsymbol{N}$ 矩阵称为 $\boldsymbol{A}$ 的广义逆矩阵，记作 $\boldsymbol{N}=\boldsymbol{A}^{-}$。如果 $\boldsymbol{A}$ 矩阵是一个 $n\times m$ 的长方形矩阵，则 $\boldsymbol{N}$ 矩阵为 $m\times n$ 阶矩阵。满足这一条件的广义逆矩阵有无穷多个。

定义下面的范数最小化指标为

$$\min_{\boldsymbol{x}}||\boldsymbol{Ax}-\boldsymbol{B}|| \tag{4-2-15}$$

则可以证明，对于一个给定的矩阵 $\boldsymbol{A}$，存在一个惟一的矩阵 $\boldsymbol{M}$ 使得下面的 3 个条件同时成立。

① $\boldsymbol{AMA}=\boldsymbol{A}$

② $\boldsymbol{MAM}=\boldsymbol{M}$

③ $\boldsymbol{AM}$ 与 $\boldsymbol{MA}$ 均为 Hermite 对称矩阵

这样的矩阵 $\boldsymbol{M}$ 称为矩阵 $\boldsymbol{A}$ 的 Moore-Penrose 广义逆矩阵，记作 $\boldsymbol{M}=\boldsymbol{A}^{+}$。从上面的 3 个条件中可以看出，第一个条件和一般广义逆的定义也是一样的，所不同的是它还要求满足第二个和第三个条件，这样就会得出惟一的广义逆矩阵了。

MATLAB 提供了求取矩阵 Moore-Penrose 广义逆的函数 pinv()。该函数的调用格式为

$\boldsymbol{M}$=pinv($\boldsymbol{A}$)	按默认精度求取 Moore-Penrose 广义逆
$\boldsymbol{M}$=pinv($\boldsymbol{A}$,ϵ)	按指定精度 ϵ 求解广义逆矩阵

其中，ϵ 为判 0 用误差限，如果省略此参数，则判 0 用误差限选用机器的精度 eps，这时将返回 $\boldsymbol{A}$ 的 Moore-Penrose 广义逆矩阵 $\boldsymbol{M}$。如果 $\boldsymbol{A}$ 矩阵为非奇异方阵，则该函数得出的结果就是矩阵的逆阵，但这样求解的速度将明显慢于 inv() 函数。

【例 4-18】考虑例 4-7 中给出的奇异矩阵 $\boldsymbol{A}$，例 4-16 中用符号运算工具箱中 inv() 函数仍不能获得问题的解析解，因为解析解不存在。所以这里将考虑 Moore-Penrose 广义逆矩阵的求解。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];
   B=pinv(A)
B =
```

```
   0.10110294117647  -0.07389705882353  -0.06139705882353   0.06360294117647
  -0.03639705882353   0.03860294117647   0.02610294117647   0.00110294117647
   0.01360294117647  -0.01139705882353  -0.02389705882353   0.05110294117647
  -0.04889705882353   0.07610294117647   0.08860294117647  -0.08639705882353
```

从得出的结果可见，这时 $\boldsymbol{B}$ 矩阵的值就没有例 4-16 那么悬殊了。$\boldsymbol{A}\boldsymbol{A}^{+}$ 的结果如下：

```
>> A*B
ans =
   0.95000000000000  -0.15000000000000   0.15000000000000   0.05000000000000
  -0.15000000000000   0.55000000000000   0.45000000000000   0.15000000000000
   0.15000000000000   0.45000000000000   0.55000000000000  -0.15000000000000
   0.05000000000000   0.15000000000000  -0.15000000000000   0.95000000000000
```

这个结果就不再是单位阵了，因为不存在一个 $\boldsymbol{A}^{+}$ 能使它成为单位阵。这样得出的 $\boldsymbol{A}^{+}$ 应该能使得式 (4-2-15) 中的范数取最小值。现在检验 Moore-Penrose 广义逆的 3 个条件如下：

```
>> norm(A*B*A-A)
ans =
    3.492484053793607e-014
>> norm(B*A*B-B)
ans =
    1.142683874961994e-016
>> [norm(A*B-(A*B)'), norm(B*A-(B*A)')]
ans =
    1.0e-014 *
    0.13532272028157    0.17379885003143
```

由此证实得出的逆矩阵确实是原矩阵的 Moore-Penrose 广义逆。现在对得出的 $\boldsymbol{B}$ 再求一次 Moore-Penrose 广义逆，则可看出，$(\boldsymbol{A}^{+})^{+}=\boldsymbol{A}$。

```
>> pinv(B)
ans =
  15.99999999999998   1.99999999999999   3.00000000000000  12.99999999999999
   5.00000000000000  11.00000000000000  10.00000000000000   8.00000000000000
   8.99999999999998   6.99999999999999   6.00000000000000  11.99999999999999
   4.00000000000000  13.99999999999999  14.99999999999999   1.00000000000000
```

【例 4-19】考虑一个给定的长方形矩阵

$$\boldsymbol{A}=\begin{bmatrix}6&1&4&2&1\\3&0&1&4&2\\-3&-2&-5&8&4\end{bmatrix}$$

请对该矩阵进行基本分析，例如获得矩阵的秩、Moore-Penrose 广义逆等，并分析得出的广义逆矩阵性质。

【求解】 可以给出下面的语句对该矩阵进行分析，得出矩阵为非满秩矩阵的结论。

```
>> A=[6,1,4,2,1; 3,0,1,4,2; -3,-2,-5,8,4]; rank(A)
   ans =
         2
```

由于 A 矩阵为奇异矩阵，所以应该使用 pinv() 函数求取矩阵的 Moore-Penrose 广义逆，并可用通过下面的检验语句对 Moore-Penrose 广义逆的条件逐一验证，证实该广义逆矩阵确实满足条件。

```
>> iA = pinv(A) % 非满秩矩阵的广义逆
iA =
    0.07302474062251   0.04130087789306  -0.02214684756584
    0.01077414205906   0.00199521149242  -0.01556264964086
    0.04588986432562   0.01775738228252  -0.03850758180367
    0.03272146847566   0.04309656823623   0.06384676775738
    0.01636073423783   0.02154828411812   0.03192338387869
>> norm(iA*A*iA-iA)   % 测试关系式 A+AA+ = A+
ans =
    1.026361262660782e-016
>> norm(A*iA*A-A) % 测试关系式 AA+A = A
ans =
    8.114540715241524e-015
>> norm(iA*A-A'*iA')  % 测试 A+A 的对称性
ans =
    3.909842920671970e-016
>> norm(A*iA-iA'*A')  % 测试 AA+ 的对称性
ans =
   1.665334536937735e-016
```

4.2.3 矩阵的特征值问题

4.2.3.1 一般矩阵的特征值与特征向量

对一个矩阵 A 来说，如果存在一个非零的向量 x，且有一个标量 λ 满足

$$Ax = \lambda x \tag{4-2-16}$$

则称 λ 为 A 矩阵的一个特征值，而 x 称为对应于特征值 λ 的特征向量。严格说来，x 应该称为 A 的右特征向量。如果矩阵 A 的特征值不包含重复的值，则对应的各个特征向量为线性无关的，这样由各个特征向量可以构成一个非奇异的矩阵。如果用它对原始矩阵作相似变换，则可以得出一个对角矩阵。矩阵的特征值与特征向量由 MATLAB 提供的函数 eig() 可以容易地求出。该函数的调用格式为

```
d=eig(A)          只求解特征值
[V, D]=eig(A)     求解特征值和特征向量
```

其中，$\boldsymbol{d}$ 特征值构成的向量，$\boldsymbol{D}$ 为一个对角矩阵，其对角线上的元素为矩阵 $\boldsymbol{A}$ 的特征值，而每个特征值对应的 $\boldsymbol{V}$ 矩阵的列为该特征值的特征向量，该矩阵是一个满秩矩阵。MATLAB 的矩阵特征值的结果满足 $\boldsymbol{AV}=\boldsymbol{VD}$，且每个特征向量各元素的平方和 (即 2 范数) 均为 1。如果调用该函数时只给出一个返回变量，则将只返回矩阵 $\boldsymbol{A}$ 的特征值。即使 $\boldsymbol{A}$ 为复数矩阵，也照样可以由 eig() 函数得出其特征值与特征向量矩阵的。

前面介绍的矩阵特征多项式的根和特征值是同样的概念，所以若精确已知矩阵的特征多项式系数，则可以调用 roots() 函数来计算矩阵的特征值。

矩阵特征值的求解算法是多种多样的，最常用的有求解实对称矩阵特征值与特征向量的 Jacobi 算法，有原点平移 QR 分解法与两步 QR 算法。矩阵的特征值与特征向量的求解有许多标准的子程序或程序库可以直接调用，如著名的 EISPACK 软件包[10, 42]等。MATLAB 中的 eig() 函数是基于两步 QR 算法实现的，该函数也同样可以求解复数矩阵的特征值与特征向量矩阵。当矩阵含有重特征根时，特征向量矩阵可能趋于奇异，所以在使用此函数时应该注意。

【例 4-20】 求出例 4-7 中给出的矩阵 $\boldsymbol{A}$ 的特征值与特征向量矩阵。

【求解】 可以调用 eig() 函数直接获得矩阵 $\boldsymbol{A}$ 的特征值。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1]; eig(A)
ans =
   33.99999999999999
    8.94427190999916
   -8.94427190999916
    0.00000000000000
```

符号运算工具箱中也提供了 eig() 函数，理论上可以求解任意高阶矩阵的精确特征根，对于给定的 $\boldsymbol{A}$ 矩阵，可以由下面的命令求出特征根的精确解为

```
>> eig(sym(A))
ans =
[          0]
[         34]
[  4*5^(1/2)]
[ -4*5^(1/2)]
```

如果想求出高精度的数值解，则可以给出如下命令

```
>> vpa(ans,70)
ans =
[                                                                       0.]
[                                                                      34.]
[  8.944271909999158785636694674925104941762473438446102897083588981642084]
```

```
[ -8.9442719099991587856366946749251049417624734384461028970835889816420084]
```

对于上面同样的 A 矩阵，可以通过下面的语句同时求出矩阵的特征值和特征向量为

```
>> [v, d] = eig(A)
v =
   -0.50000000000000  -0.82360679774998   0.37639320225002  -0.22360679774998
   -0.50000000000000   0.42360679774998   0.02360679774998  -0.67082039324994
   -0.50000000000000   0.02360679774998   0.42360679774998   0.67082039324994
   -0.50000000000000   0.37639320225002  -0.82360679774998   0.22360679774998
d =
   33.99999999999999                  0                  0                  0
                   0   8.94427190999916                  0                  0
                   0                  0  -8.94427190999916                  0
                   0                  0                  0   0.00000000000000
```

同样，用符号运算工具箱中的 eig() 函数也可以求解出特征值和特征向量矩阵的解析解。如下：

```
>> [v,d]=eig(sym(A))
v =
    [          -1,            1, -8*5^(1/2)-17,  8*5^(1/2)-17]
    [          -3,            1,    4*5^(1/2)+9,  -4*5^(1/2)+9]
    [           3,            1,             1,             1]
    [           1,            1,    4*5^(1/2)+7,  -4*5^(1/2)+7]
d =
    [           0,            0,             0,             0]
    [           0,           34,             0,             0]
    [           0,            0,     4*5^(1/2),             0]
    [           0,            0,             0,    -4*5^(1/2)]
```

可见，在前面的例子中两次调用了 eig() 函数，但由于返回参数个数不一致，所以前面的调用返回矩阵 A 的特征值与特征向量，而后面的调用只返回了矩阵 A 的特征值而不返回特征向量矩阵。另外，返回特征值的格式也因返回变量个数不同而不同。

如果一个矩阵包含重特征根，则理论上矩阵 V 将为奇异矩阵。但因为 MATLAB 数值运算出现的误差，不一定能精确计算出矩阵的重根，这样将得出接近奇异的 V 矩阵。

若想由 C 或 Fortran 语句从最底层编程，则需要编写相当大的程序才可以完成特征值和特征向量的计算，例如，EISPACK[42] 中提供的子程序源程序有 500 多条。

4.2.3.2 矩阵的广义特征向量问题

若某矩阵 A 含有重特征根，则必定会使得特征向量矩阵为奇异矩阵，这会约束特征向量矩阵的应用。为了保证特征向量矩阵非奇异，需要引入广义特征向量的问题。假设

存在一个标量 λ 和一个非零向量 $\boldsymbol{x}$，使得

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x} \tag{4-2-17}$$

成立，其中 $\boldsymbol{B}$ 矩阵为对称正定矩阵，则 λ 称为广义特征值，而 $\boldsymbol{x}$ 向量称为广义特征向量。MATLAB 还提供了求取广义特征值的方法。事实上，普通的矩阵特征值问题可以看成是广义特征值问题的一个特例，因为若假定 $\boldsymbol{B}=\boldsymbol{I}$ 为单位阵，则式 (4-2-17) 中的形式可以直接转化成普通矩阵特征值问题。

如果 $\boldsymbol{B}$ 矩阵为一个非奇异方阵，则上面的方程可以容易地转换成一般矩阵 $\boldsymbol{B}^{-1}\boldsymbol{A}$ 的特征值问题

$$\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x} \tag{4-2-18}$$

即 λ 和 $\boldsymbol{x}$ 分别为 $\boldsymbol{B}^{-1}\boldsymbol{A}$ 矩阵的特征值和特征向量。但一般情况下不能随便假设 $\boldsymbol{B}$ 阵为非奇异的方阵，所以文献 [31] 中给出了广义特征值问题的 QZ 算法。在 MATLAB 中给出的 eig() 函数可以直接用来求取矩阵的广义特征值和特征向量，这时的调用格式为

d=eig(*A*,*B*)	求解广义特征值
[*V*,*D*]=eig(*A*,*B*)	求解广义特征值和特征向量

这一函数可以直接得出矩阵的广义特征值向量 $\boldsymbol{d}$，也可以返回一个特征向量矩阵 $\boldsymbol{V}$ 及一个对角型特征值矩阵 $\boldsymbol{D}$，满足 $\boldsymbol{AV}=\boldsymbol{BVD}$。值得指出的是，该函数可以求解 $\boldsymbol{B}$ 矩阵为奇异矩阵时的广义特征值问题。

【例 4-21】假设给出如下的矩阵

$$\boldsymbol{A}=\begin{bmatrix}5&7&6&5\\7&10&8&7\\6&8&10&9\\5&7&9&10\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}2&6&-1&-2\\5&-1&2&3\\-3&-4&1&10\\5&-2&-3&8\end{bmatrix}$$

请求出 $\boldsymbol{A},\boldsymbol{B}$ 矩阵的广义特征值与特征向量矩阵。

【求解】使用下列命令可以求出矩阵的广义特征值和特征向量。

```
>> A=[5,7,6,5; 7,10,8,7; 6,8,10,9; 5,7,9,10];
   B=[2,6,-1,-2; 5,-1,2,3; -3,-4,1,10; 5,-2,-3,8];
   [V, D] = eig(A, B)
V =
   0.2224 - 0.0000i  0.5157 - 0.0188i -0.0188 + 0.5157i  0.8328 + 0.0000i
   0.5985 - 0.0000i -0.1244 + 0.1375i  0.1375 - 0.1244i -0.5072 + 0.0000i
   0.4800 - 0.0000i -0.3400 - 0.5588i -0.5588 - 0.3400i -0.1929 - 0.0000i
   0.6016 - 0.0000i  0.0603 + 0.5175i  0.5175 + 0.0603i  0.1098 + 0.0000i
D =
   4.7564 + 0.0000i        0                  0                 0
        0           0.0471 + 0.1750i          0                 0
        0                  0           0.0471 - 0.1750i         0
```

```
         0                  0                  0            -0.0037
>> norm(A*V-B*V*D)
ans =
    1.5783e-014
```

再考虑 B 矩阵奇异的情形。例如，B 矩阵为 4×4 魔方矩阵，即 B=magic(4)，则可以用下面的 MATLAB 语句计算出 (A,B) 矩阵的广义特征值为

```
>> B=magic(4); eig(A,B)
ans =
   0.91539141931817
   0.00096432883641
  -0.14118127158429
                Inf
```

符号运算工具箱中的 eig() 函数不支持广义特征值的运算。

4.3 矩阵的基本变换

4.3.1 矩阵的相似变换与正交矩阵

对某方阵 A 来说，如果存在一个非奇异的 B 矩阵，则可以通过下面的方式对原 A 矩阵进行变换。

$$X = B^{-1}AB \tag{4-3-1}$$

这样的变换称为相似变换，而 B 称为相似变换矩阵。相似变换后，X 矩阵的秩、迹、行列式和特征值等均不发生变化，其值和 A 矩阵完全一致。通过适当选择变换矩阵 B，就能有目的地将任意给定的 A 矩阵相似变换成特殊的矩阵表示形式，而不改变原来 A 的重要性质。

对于一类特殊的相似变换矩阵 T，如果它本身满足 $T^{-1}=T^*$，其中 T^* 为 T 的 Hermite 共轭转置矩阵，则称 T 为正交矩阵，并将之记为 $Q=T$。可见，正交矩阵 Q 满足条件

$$Q^*Q=I,\ 且\ QQ^*=I \tag{4-3-2}$$

其中，I 为 $n\times n$ 的单位阵。

MATLAB 中提供了求取正交矩阵的函数 orth()，该函数的调用格式为

Q=orth(A)

该函数能求出 A 矩阵的正交基矩阵 Q。若 A 为非奇异矩阵，则得出的正交基矩阵 Q 满足式 (4-3-2) 的条件。若 A 为奇异矩阵，则得出的矩阵 Q 的列数即为 A 矩阵的秩，且满足 $Q^*Q=I$，而不满足 $QQ^*=I$。

【例 4-22】 求出下面给出的 $\boldsymbol{A}$ 矩阵的正交矩阵。

$$\boldsymbol{A}=\begin{bmatrix}5&9&8&3\\0&3&2&4\\2&3&5&9\\3&4&5&8\end{bmatrix}$$

【求解】 一个已知矩阵的正交矩阵可以用 orth() 函数直接得出，并可以验证满足正交矩阵的性质。

```
>> A=[5,9,8,3; 0,3,2,4; 2,3,5,9; 3,4,5,8];
   Q=orth(A)
Q =
  -0.61967134014030   0.77381388090439  -0.02618727464494  -0.12858357039325
  -0.25484757692751  -0.15505965525406   0.94903028305249   0.10173857526022
  -0.51978106566308  -0.52982004408860  -0.15628278874693  -0.65168554886662
  -0.52998847772481  -0.31057897854511  -0.27245447039361   0.74055484140430
>> norm(Q'*Q-eye(4))
ans =
     4.639540927399602e-016
>> norm(Q*Q'-eye(4))
ans =
     4.926988676171614e-016
```

【例 4-23】 重新考虑例 4-7 中给出的奇异矩阵 $\boldsymbol{A}$，试求出其正交基矩阵，并验证其性质。

【求解】 可以通过下面的 MATLAB 语句求取并检验其正交基矩阵。注意，因为 $\boldsymbol{A}$ 为奇异矩阵，故得出的 $\boldsymbol{Q}$ 为长方形矩阵。

```
>> A=[16,2,3,13; 5,11,10,8; 9,7,6,12; 4,14,15,1];
   Q=orth(A)
Q =
  -0.50000000000000   0.67082039324994   0.50000000000000
  -0.50000000000000  -0.22360679774998  -0.50000000000000
  -0.50000000000000   0.22360679774998  -0.50000000000000
  -0.50000000000000  -0.67082039324994   0.50000000000000
>> norm(Q'*Q-eye(3))
ans =
     1.014021488513403e-015
```

4.3.2 矩阵的三角分解和 Cholesky 分解

4.3.2.1 一般矩阵的三角分解

矩阵的三角分解又称为 LU 分解，它的目的是将一个矩阵分解成一个下三角矩阵 $\boldsymbol{L}$

和一个上三角矩阵 $\boldsymbol{U}$ 的乘积，亦即 $\boldsymbol{A}=\boldsymbol{LU}$, 其中 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 矩阵可以分别写成

$$\boldsymbol{L}=\begin{bmatrix}1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1\end{bmatrix},\quad \boldsymbol{U}=\begin{bmatrix}u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn}\end{bmatrix} \tag{4-3-3}$$

由这两个矩阵可以简单地写出一个矩阵 $\boldsymbol{A}^F$, 其中

$$\boldsymbol{A}^F=\boldsymbol{L}+\boldsymbol{U}-\boldsymbol{I}=\begin{bmatrix}u_{11} & u_{12} & \cdots & u_{1n} \\ l_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & u_{nn}\end{bmatrix} \tag{4-3-4}$$

这样产生的矩阵与原来的 $\boldsymbol{A}$ 矩阵的关系可以写成

$$\begin{array}{llll}a_{11}=u_{11}, & a_{12}=u_{12}, & \cdots & a_{1n}=u_{1n} \\ a_{21}=l_{21}u_{11}, & a_{22}=l_{21}u_{12}+u_{22}, & \cdots & u_{2n}=l_{21}u_{1n}+u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}=l_{n1}u_{11}, & a_{n2}=l_{n1}u_{12}+l_{n2}u_{22}, & \cdots & a_{nn}=\displaystyle\sum_{k=1}^{n-1}l_{nk}u_{kn}+u_{nn}\end{array} \tag{4-3-5}$$

由式 (4-3-5) 可以立即得出求取 l_{ij} 和 u_{ij} 的递推计算公式

$$l_{ij}=\frac{a_{ij}-\displaystyle\sum_{k=1}^{j-1}l_{ik}u_{kj}}{u_{jj}},\ (j<i),\ \text{及}\ u_{ij}=a_{ij}-\sum_{k=1}^{i-1}l_{ik}u_{kj},\ (j\geqslant i) \tag{4-3-6}$$

该公式的递推初值为

$$u_{1i}=a_{1i},\ i=1,2,\cdots,n \tag{4-3-7}$$

注意，在上述的算法中并未对主元素进行任何选取，因此该算法并不一定数值稳定，因为在运算过程中 0 或很小的数值可能被用作除数。在 MATLAB 中也给出了基于主元素的矩阵 LU 分解函数 `lu()`，该函数的调用格式为

```
[L, U] = lu(A)         LU 分解，A = LU
[L, U, P] = lu(A)      P 为置换矩阵，A = P^-1 LU
```

其中，$\boldsymbol{L}, \boldsymbol{U}$ 分别为变换后的下三角和上三角矩阵。在 MATLAB 的 `lu()` 函数中考虑了主元素选取的问题，所以该函数一般会给出可靠的结果。由该函数得出的下三角矩阵 $\boldsymbol{L}$ 并不一定是一个真正的下三角矩阵，因为选取它可能进行了一些元素行的交换，这样主对角线的元素可能不是 1, 而在矩阵 $\boldsymbol{L}$ 内存在一个惟一的如式 (4-2-1) 中定义的置换，其各个元素的值均是 1。如果想获得有关换行信息，则可以由后一种格式调用 `lu()` 函数，这时 $\boldsymbol{P}$ 为单位阵变换出的置换矩阵，$\boldsymbol{A}$ 矩阵可以分解成 $\boldsymbol{A}=\boldsymbol{P}^{-1}\boldsymbol{LU}$。

【例 4-24】再考虑例 4-7 中矩阵的 LU 分解问题。分别用两种方法调用 MATLAB 中的 lu() 函数，则可以得出的不同结果。

【求解】先输入 $\boldsymbol{A}$ 矩阵，并求出三角分解矩阵。

```
>> A=[16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];
   [L1,U1]=lu(A)
L1 =
   1.00000000000000                  0                  0                  0
   0.31250000000000   0.76851851851852   1.00000000000000                  0
   0.56250000000000   0.43518518518519   1.00000000000000   1.00000000000000
   0.25000000000000   1.00000000000000                  0                  0
U1 =
  16.00000000000000   2.00000000000000   3.00000000000000  13.00000000000000
                  0  13.50000000000000  14.25000000000000  -2.25000000000000
                  0                  0  -1.88888888888889   5.66666666666667
                  0                  0                  0   0.00000000000000
```

可见，这样得出的 $\boldsymbol{L}_1$ 矩阵并非下三角矩阵，这是因为在分解过程中采用了主元素交换的方法。现在考虑 lu() 函数的另一种调用方法。

```
>> [L, U, P] = lu(A)
L =
   1.00000000000000                  0                  0                  0
   0.25000000000000   1.00000000000000                  0                  0
   0.31250000000000   0.76851851851852   1.00000000000000                  0
   0.56250000000000   0.43518518518519   1.00000000000000   1.00000000000000
U =
  16.00000000000000   2.00000000000000   3.00000000000000  13.00000000000000
                  0  13.50000000000000  14.25000000000000  -2.25000000000000
                  0                  0  -1.88888888888889   5.66666666666667
                  0                  0                  0   0.00000000000000
P =
      1     0     0     0
      0     0     0     1
      0     1     0     0
      0     0     1     0
```

注意，这里得出的 $\boldsymbol{P}$ 矩阵不是一个单位矩阵，而是单位矩阵的置换矩阵。结合得出的 $\boldsymbol{L}_1$ 矩阵可以看出，$\boldsymbol{P}$ 矩阵的 $p_{2,4}=1$，表明需要将 $\boldsymbol{L}_1$ 矩阵的第 4 行换到第 2 行，$p_{3,2}=p_{4,3}=1$ 表明需要将 $\boldsymbol{L}_1$ 的第 2 行换至第 3 行，将原第 3 行换至第 4 行，这样就可以得出一个真正的下三角矩阵 $\boldsymbol{L}$ 了。将 $\boldsymbol{P},\boldsymbol{L},\boldsymbol{U}$ 代入并检验，可以精确地还原 $\boldsymbol{A}$ 矩阵。

```
>> inv(P)*L*U
```

```
ans =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

4.3.2.2 对称矩阵的三角分解 —— Cholesky 分解

如果 $\boldsymbol{A}$ 为对称矩阵，利用对称矩阵的特点则可以用 LU 分解的方法对之进行分解，这样可以将原来矩阵 $\boldsymbol{A}$ 分解成

$$\boldsymbol{A}=\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D}=\begin{bmatrix} d_{11} & & & \\ d_{21} & d_{22} & & \\ \vdots & \vdots & \ddots & \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{bmatrix}\begin{bmatrix} d_{11} & d_{21} & \cdots & d_{n1} \\ & d_{22} & \cdots & d_{n2} \\ & & \ddots & \vdots \\ & & & d_{nn} \end{bmatrix} \tag{4-3-8}$$

如果利用对称矩阵的性质，则 LU 分解可以简化成

$$d_{ii}=\sqrt{a_{ii}-\sum_{k=1}^{i-1}d_{ik}^2},\ \ d_{ij}=\frac{a_{ij}-\sum\limits_{k=1}^{j-1}d_{ik}d_{jk}}{l_{jj}},\ (j<i) \tag{4-3-9}$$

该分解算法又称为对称矩阵是 Cholesky 分解算法。

MATLAB 提供了 `chol()` 函数来求取矩阵的 Cholesky 分解矩阵 $\boldsymbol{D}$，其结果为一个下三角矩阵。该函数的调用格式为

```
D = chol(A)
```

【例 4-25】考虑一个对称的 4 阶 $\boldsymbol{A}$ 矩阵

$$\boldsymbol{A}=\begin{bmatrix} 9 & 3 & 4 & 2 \\ 3 & 6 & 0 & 7 \\ 4 & 0 & 6 & 0 \\ 2 & 7 & 0 & 9 \end{bmatrix}$$

用下面的语句可以对之进行 Cholesky 分解，得出 $\boldsymbol{D}$ 矩阵。

```
>> A=[9,3,4,2; 3,6,0,7; 4,0,6,0; 2,7,0,9];
   D=chol(A)
D =
   3.00000000000000   1.00000000000000   1.33333333333333   0.66666666666667
                  0   2.23606797749979  -0.59628479399994   2.83235277149973
                  0                  0   1.96638416050035   0.40683810217249
                  0                  0                  0   0.60647843486312
```

4.3.2.3 正定、正规矩阵的定义与判定

正定矩阵是在对称矩阵基础上建立起来的概念。在介绍该概念之前，先给出主子行列式定义。假设对称矩阵 $\boldsymbol{A}$ 可以写成

$$\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1,n} \\ a_{12} & a_{22} & a_{23} & \cdots & a_{2,n} \\ a_{13} & a_{23} & a_{33} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1,n} & a_{2,n} & a_{3,n} & \cdots & a_{n,n} \end{bmatrix} \tag{4-3-10}$$

则左上角的各个子矩阵的行列式称为主子行列式。如果一个对称矩阵所有的主子行列式均为正数，则称该矩阵为正定矩阵。

相应地，可以引入对称矩阵的半正定矩阵的概念，如果主子行列式均为非负的数值，则称为半正定矩阵。MATLAB 的函数 `chol()` 还可以用来判定矩阵的正定性。该函数的另一种调用格式为

```
[D,p]=chol(A)
```

式中对正定的 $\boldsymbol{A}$ 矩阵来说，返回的 $p=0$。所以可以利用这个性质来判定一个对称矩阵是否为正定矩阵。对非正定矩阵，则返回一个正的 p 值，$p-1$ 为 $\boldsymbol{A}$ 矩阵中正定的子矩阵的阶次，亦即 $\boldsymbol{D}$ 将为 $(p-1)$ 阶方阵。

如果复数方阵满足

$$\boldsymbol{A}^*\boldsymbol{A}=\boldsymbol{A}\boldsymbol{A}^* \tag{4-3-11}$$

则其中 $\boldsymbol{A}^*$ 为 $\boldsymbol{A}$ 的 Hermite 转置，亦即共轭转置，这样该矩阵称为正规矩阵。判定正规矩阵由定义可以直接完成

```
norm(A'*A-A*A')<ϵ
```

如果得出的结果为 1，则 $\boldsymbol{A}$ 为正规矩阵。

【例 4-26】试判定下面的对称矩阵 $\boldsymbol{A}$ 是否为正定矩阵，并求出其 Cholesky 分解矩阵。

$$\boldsymbol{A}=\begin{bmatrix} 7 & 5 & 5 & 8 \\ 5 & 6 & 9 & 7 \\ 5 & 9 & 9 & 0 \\ 8 & 7 & 0 & 1 \end{bmatrix}$$

【求解】用下面的语句可以对 $\boldsymbol{A}$ 矩阵进行分解，得出 $\boldsymbol{D}$ 矩阵，并求出正定的阶次为 2，从而说明该矩阵并非正定矩阵，因为得出的 $\boldsymbol{D}$ 矩阵不满秩，且 $p\neq 0$。

```
>> A=[7,5,5,8; 5,6,9,7; 5,9,9,0; 8,7,0,1];
   [D,p]=chol(A)
D =
   2.64575131106459   1.88982236504614
                  0   1.55838744494796
p =
```

```
        3
>> norm(A*A'-A'*A)<eps  % 判定正规矩阵
ans =
        1
```

非对称矩阵也可以调用 chol() 函数，但结果是错误的，它将首先将给定的矩阵强制按上三角子矩阵转换成对称矩阵。在严格的数学意义下，非对称矩阵是没有 Cholesky 分解的。

4.3.3 矩阵的 Jordan 变换

对含有重特征根矩阵 $\boldsymbol{A}$ 通常不能直接分解成对角矩阵，而用纯特征根求解方法必定能使矩阵的特征向量矩阵 $\boldsymbol{V}$ 含有若干相同的列，使得该矩阵为奇异矩阵。

【例 4-27】分别用数值方法和解析方法求出下面矩阵的特征值与特征向量矩阵。

$$\boldsymbol{A}=\begin{bmatrix}-71 & -65 & -81 & -46\\ 75 & 89 & 117 & 50\\ 0 & 4 & 8 & 4\\ -67 & -121 & -173 & -58\end{bmatrix}$$

【求解】用 MATLAB 语言中的数值算法可以求出该矩阵的特征根为

```
>> A=[-71,-65,-81,-46; 75,89,117,50; 0,4,8,4; -67,-121,-173,-58];
   D=eig(A)    % 特征向量矩阵从略
D =
    -8.00643327508056
    -7.99999975484467 + 0.00643302994896i
    -7.99999975484467 - 0.00643302994896i
    -7.99356721523009
```

其实，该矩阵的特征根为位于 -8 的 4 重根，所以用数值解方法得出的特征根有很大的误差，故在这样的问题上不适合采用数值算法。

如果采用解析算法，则可以得出如下结果：

```
>> [v,d]=eig(sym(A))
v =
   [ 17/8]
   [ -13/8]
   [     1]
   [ -19/8]
d =
   [ -8,  0,  0,  0]
   [  0, -8,  0,  0]
   [  0,  0, -8,  0]
   [  0,  0,  0, -8]
```



可见，该矩阵特征根均为 -8，为 4 重根，故而得出的特征向量矩阵实际上是奇异矩阵，因为 4 列均相同，所以只保留了一列。

由于 MATLAB 语言、其他数值运算语言、软件包本身及数值算法的缺陷，均在计算内核中使用 double 型数据结构，导致对某些有重根矩阵的计算中间过程出现误差，所以在纯数值运算上将有很大的误差，即使使用现在最好的数值运算方法，如果不突破双精度的数据结构也是难以避免的。

为解决这样的问题，需要使用符号运算工具箱中的 jordan() 函数来分解出 Jordan 标准型，并求出非奇异的广义特征向量矩阵。该函数的调用格式为

J=jordan(A)	只返回 Jordan 矩阵 J
[V,J]=jordan(A)	返回 Jordan 矩阵 J 和广义特征向量矩阵 V

有了广义特征向量矩阵 V，则 Jordan 标准型可以由 $J = V^{-1}AV$ 变换出来。注意，Jordan 矩阵主对角线为矩阵的特征根，次主对角线为 1。

【例 4-28】试对例 4-27 中给出的矩阵进行 Jordan 分解。

【求解】符号矩阵的 Jordan 分解可以用 jordan() 函数直接进行分解，得出所需的矩阵。

```
>> A=[-71,-65,-81,-46; 75,89,117,50; 0,4,8,4; -67,-121,-173,-58];
   [V,J]=jordan(sym(A))
V =
   [ -18496,   2176,    -63,       1]
   [  14144,   -800,     75,       0]
   [  -8704,     32,      0,       0]
   [  20672,  -1504,    -67,       0]
J =
   [ -8,  1,  0,  0]
   [  0, -8,  1,  0]
   [  0,  0, -8,  1]
   [  0,  0,  0, -8]
```

这样得出的 V 矩阵就是满秩的矩阵了，对它求逆，就可以实现用普通数值运算难以实现的功能。该问题将在后面矩阵函数的例子中演示。

【例 4-29】已知具有复数特征根的矩阵 A 由下式给出，

$$A = \begin{bmatrix} -10 & -14 & -17 & -9 \\ 10 & 16 & 18 & 10 \\ 0 & -1 & 0 & 0 \\ -9 & -15 & -17 & -10 \end{bmatrix}$$

试求出其 Jordan 分解。

【求解】复数特征根矩阵的 Jordan 标准型及广义特征向量矩阵问题也可以由 jordan() 函数求取。可以用下面的语句得出相应的特征向量矩阵与 Jordan 矩阵为

```
>> A=[-10,-14,-17,-9; 10,16,18,10; 0,-1,0,0; -9,-15,-17,-10];
```

```
    [V,J]=jordan(sym(A))
 V =
    [ -11+2*i, -11-2*i,      -5,      23]
    [   5-5*i,   5+5*i,       0,     -10]
    [       5,       5,       0,     -10]
    [  -6+7*i,  -6-7*i,       5,      12]
 J =
    [ -1+i,     0,     0,     0]
    [    0, -1-i,     0,     0]
    [    0,     0,    -1,     1]
    [    0,     0,     0,    -1]
```

4.3.4 矩阵的奇异值分解

矩阵的奇异值也可以看成是矩阵的一种测度。对任意的 $n \times m$ 矩阵 $\boldsymbol{A}$ 来说，总有

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} \geqslant 0,\ \ \boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} \geqslant 0 \tag{4-3-12}$$

且在理论上有

$$\mathrm{rank}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}) = \mathrm{rank}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}) = \mathrm{rank}(\boldsymbol{A}) \tag{4-3-13}$$

进一步可以证明，$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 与 $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ 有相同的非负特征值 λ_i, 在数学上把这些非负的特征值的平方根称作矩阵 $\boldsymbol{A}$ 的奇异值，记 $\sigma_i(\boldsymbol{A}) = \sqrt{\lambda_i(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})}$。

【例 4-30】现在考虑一个演示例子，假设矩阵 $\boldsymbol{A}$ 为

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ \mu & 0 \\ 0 & \mu \end{bmatrix}$$

其中 $\mu = 5\mathrm{eps}$，试根据式 (4-3-13) 求取 $\boldsymbol{A}$ 矩阵的秩。

【求解】显然，$\boldsymbol{A}$ 矩阵的秩为 2。用 MATLAB 运算也将得出同样的结论。

```
>> A=[1 1; 5*eps,0; 0,5*eps]; rank(A)
ans =
      2
```

考虑式 (4-3-13) 中给出的方法计算矩阵 $\boldsymbol{A}$ 的秩。例如，用 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 来求解 $\boldsymbol{A}$ 的秩，则可以得出

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \begin{bmatrix} 1+\mu^2 & 1 \\ 1 & 1+\mu^2 \end{bmatrix}$$

在双精度数值运算中，由于 μ^2 为 10^{-30} 级数值，所以加到 1 上事实上就已经被忽略了，所以 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 矩阵将退化成幺矩阵，再求其秩显然为 1，从而可以断定原矩阵 $\boldsymbol{A}$ 的秩为 1，这与实际矛盾，故对这样的问题应该引入一个新的量作为矩阵的测度，即需要引入奇异值的概念。

假设 $\boldsymbol{A}$ 矩阵为 $n\times m$ 矩阵，则 $\boldsymbol{A}$ 矩阵可以分解为

$$\boldsymbol{A}=\boldsymbol{L}\boldsymbol{A}_1\boldsymbol{M} \tag{4-3-14}$$

其中，$\boldsymbol{L}$ 和 $\boldsymbol{M}$ 为正交矩阵，$\boldsymbol{A}_1=\mathrm{diag}(\sigma_1,\cdots,\sigma_n)$ 为对角矩阵，其对角元素满足不等式 $\sigma_1\geqslant\sigma_2\geqslant\cdots\geqslant\sigma_n\geqslant 0$。如果存在 $\sigma_n=0$，则原矩阵 $\boldsymbol{A}$ 为奇异矩阵，这时矩阵 $\boldsymbol{A}$ 的秩应该等于矩阵 $\boldsymbol{A}_1$ 中非 0 对角元素的个数。

MATLAB 提供了直接求取矩阵奇异值分解的函数 svd()，其调用方式为

```
S = svd(A)              % 只计算矩阵的奇异值
[L, A1, M] = svd(A)     % 矩阵奇异值与变换矩阵
```

其中，$\boldsymbol{A}$ 为原始矩阵，返回的 $\boldsymbol{A}_1$ 为对角矩阵，而 $\boldsymbol{L}$ 和 $\boldsymbol{M}$ 均为正交变换矩阵，并满足 $\boldsymbol{A}=\boldsymbol{L}\boldsymbol{A}_1\boldsymbol{M}^{\mathrm{T}}$。

矩阵的奇异值大小通常决定矩阵的性态，如果矩阵的奇异值变化特别大，则矩阵中某个元素有一个微小的变化将严重影响到原矩阵的参数，这样的矩阵又称为病态矩阵或坏条件矩阵，而在矩阵存在趋于 0 的奇异值时称为奇异矩阵。矩阵最大奇异值 $\sigma_{\max}$ 和最小奇异值 $\sigma_{\min}$ 的比值又称为该矩阵的条件数，记作 $\mathrm{cond}(\boldsymbol{A})$，即 $\mathrm{cond}(\boldsymbol{A})=\sigma_{\max}/\sigma_{\min}$，矩阵的条件数越大，则对元素变化越敏感。矩阵的最大奇异值和最小奇异值还分别记作 $\bar{\sigma}(\boldsymbol{A})$ 和 $\underline{\sigma}(\boldsymbol{A})$。在 MATLAB 中也提供了函数 cond(A) 来求取矩阵 $\boldsymbol{A}$ 的条件数。

【例 4-31】 试对例 4-7 中给出的 $\boldsymbol{A}$ 矩阵进行奇异值分解。

【求解】 如果调用 MATLAB 中给出的矩阵奇异值分解函数 svd()，则可以容易地求出 $\boldsymbol{L}$, $\boldsymbol{A}_1$ 和 $\boldsymbol{M}$ 矩阵，并可以容易地求出该矩阵的条件数。

```
>> [16,2,3,13; 5,11,10,8; 9,7,6,12; 4,14,15,1];
   [L, A1, M]=svd(A)
L =
  -0.50000000000000   0.67082039324994   0.50000000000000  -0.22360679774998
  -0.50000000000000  -0.22360679774998  -0.50000000000000  -0.67082039324994
  -0.50000000000000   0.22360679774998  -0.50000000000000   0.67082039324994
  -0.50000000000000  -0.67082039324994   0.50000000000000   0.22360679774998
A1 =
  33.99999999999999                  0                  0                  0
                  0  17.88854381999832                  0                  0
                  0                  0   4.47213595499958                  0
                  0                  0                  0   0.00000000000000
M =
  -0.50000000000000   0.50000000000000   0.67082039324994   0.22360679774998
  -0.50000000000000  -0.50000000000000  -0.22360679774998   0.67082039324994
  -0.50000000000000  -0.50000000000000   0.22360679774998  -0.67082039324994
  -0.50000000000000   0.50000000000000  -0.67082039324994  -0.22360679774998
```

可见该矩阵含有 0 奇异值，故原矩阵为奇异矩阵。该矩阵的条件数可以由下面的语句得出，接近于 ∞，但在双精度数值运算上有一定的误差。

```
>> cond(A)
ans =
    3.259197026649050e+016
```

如果先将 $\boldsymbol{A}$ 矩阵转换成符号矩阵，则调用 svd() 将得出更精确的奇异值分解矩阵。

【例 4-32】对于 $n \neq m$ 的矩阵 $\boldsymbol{A}$ 来说，也可以对之作奇异值分解。例如，可以对下面的长方形矩阵进行奇异值分解，并检验分解的结果。

$$\boldsymbol{A} = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}$$

【求解】使用如下命令则可以得出

```
>> A=[1,3,5,7; 2,4,6,8]; [L,A1,M]=svd(A)
L =
  -0.64142302799507  -0.76718739507218
  -0.76718739507218   0.64142302799507
A1 =
  14.26909549926148                  0                  0                  0
                  0   0.62682823241754                  0                  0
M =
  -0.15248323331020   0.82264747222566  -0.39450102228383  -0.37995913387760
  -0.34991837180796   0.42137528768458   0.24279654570436   0.80065587951006
  -0.54735351030573   0.02010310314350   0.69790997544278  -0.46143435738734
  -0.74478864880349  -0.38116908139757  -0.54620549886330   0.04073761175487
>> A2=L*A1*M'   % 还原 A 阵
A2 =
   1.00000000000000   3.00000000000000   5.00000000000000   7.00000000000000
   2.00000000000000   4.00000000000000   6.00000000000000   8.00000000000001
>> norm(A-A2)
ans =
    9.727728986738545e-015
```

对这个例子进行逆运算，即 $\boldsymbol{L}\boldsymbol{A}_1\boldsymbol{V}^{\mathrm{T}}$，则可以还原成原来的 $\boldsymbol{A}$ 矩阵。由前面的分析可见，这样得出的矩阵误差是很小的。

4.4 矩阵方程的计算机求解

4.4.1 线性方程组的计算机求解

考虑下面给出的线性代数方程

$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{B} \tag{4-4-1}$$

其中，$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 均为给定矩阵

$$\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mp} \end{bmatrix} \tag{4-4-2}$$

可以由给定的 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵构造出解的判定矩阵 $\boldsymbol{C}$

$$\boldsymbol{C}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_{11} & b_{12} & \cdots & b_{1p} \\ a_{21} & a_{22} & \cdots & a_{2n} & b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_{m1} & b_{m2} & \cdots & b_{mp} \end{bmatrix} \tag{4-4-3}$$

这样可以不加证明地给出线性方程组有解的判定定理[41]：

① 当 $m=n$，且 $\operatorname{rank}(\boldsymbol{A})=\operatorname{rank}(\boldsymbol{C})=n$ 时，方程组 (4-4-1) 有惟一解

$$\boldsymbol{x}=\boldsymbol{A}^{-1}\boldsymbol{B} \tag{4-4-4}$$

用 MATLAB 语言可以立即得出该方程的解为 `x=inv(A)*B`。但是 `inv()` 函数的调用也有值得注意之处。例如，若 $\boldsymbol{A}$ 矩阵为奇异的或接近奇异的，则利用此函数有可能产生错误的结果。

若采用符号运算工具箱，则可以直接使用 `inv()` 函数，如果能得出方程的解，则解是惟一的，如果出现错误信息，则再考虑其他的情形。

【例 4-33】 求解下面给出的线性代数方程组。

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ 1 & 3 & 2 & 4 \\ 4 & 1 & 3 & 2 \end{bmatrix}\boldsymbol{X}=\begin{bmatrix} 5 & 1 \\ 4 & 2 \\ 3 & 3 \\ 2 & 4 \end{bmatrix}$$

【求解】 上述方程可以用下面的语句直接求出，并验证其精度。

```
>> A=[1 2 3 4; 4 3 2 1; 1 3 2 4; 4 1 3 2]; B=[5 1; 4 2; 3 3; 2 4];
   x=inv(A)*B
x =
  -1.80000000000000   2.40000000000000
   1.86666666666667  -1.26666666666667
```

```
   3.86666666666667  -3.26666666666667
  -2.13333333333333   2.73333333333333
>> norm(A*x-B)
ans =
    7.473833782290161e-015
```

将上面得出的解代入原方程，可以看出误差是很小的，达到 10^{-15} 级。事实上，如果采用符号运算工具箱中的求逆函数将得出精确的解。

```
>> x1=inv(sym(A))*B
x1 =
   [   -9/5,    12/5]
   [  28/15, -19/15]
   [  58/15, -49/15]
   [ -32/15,  41/15]
>> norm(double(A*x1-B))
ans =
     0
```

② 当 $\mathrm{rank}(\boldsymbol{A})=\mathrm{rank}(\boldsymbol{C})=r<n$ 时，方程组 (4-4-1) 有无穷多解，可以构造出线性方程组的 $n-r$ 个化零向量 $\boldsymbol{x}_i, i=1,2,\cdots,n-r$，原方程组对应的齐次方程组的解 $\hat{\boldsymbol{x}}$ 可以由 $\boldsymbol{x}_i$ 的线性组合来表示，即

$$\hat{\boldsymbol{x}}=\alpha_1\boldsymbol{x}_1+\alpha_2\boldsymbol{x}_2+\cdots+\alpha_{n-r}\boldsymbol{x}_{n-r} \tag{4-4-5}$$

其中，$\alpha_i, i=1,2,\cdots,n-r$ 为任意常数。在 MATLAB 语言中可以由 `null()` 直接求出，其调用格式为

`Z=null(A)`	求解 $\boldsymbol{A}$ 矩阵的化零矩阵
`Z=null(A,'r')`	求解 $\boldsymbol{A}$ 矩阵的化零矩阵的规范形式

`null()` 函数也可以用于符号变量描述方程的解析解问题，其中 $\boldsymbol{Z}$ 的列数为 $n-r$，而各列构成的向量又称为矩阵 $\boldsymbol{A}$ 的基础解系。

求解式 (4-4-1) 中给出的非齐次方程组也是较简单的，只要能求出该方程的任意一个特解 $\boldsymbol{x}_0$，则原非齐次方程组的解为 $\boldsymbol{x}=\hat{\boldsymbol{x}}+\boldsymbol{x}_0$。其实，在 MATLAB 语言中求解该方程的一个特解并非难事，用 $\boldsymbol{x}_0$=`pinv(`$\boldsymbol{A}$`)*`$\boldsymbol{B}$ 即可求出。

【例 4-34】求解下面给出的线性代数方程组。

$$\begin{bmatrix}1&2&3&4\\2&2&1&1\\2&4&6&8\\4&4&2&2\end{bmatrix}\boldsymbol{X}=\begin{bmatrix}1\\3\\2\\6\end{bmatrix}$$

【求解】用下面语句可以输入 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵，并按式 (4-4-3) 构造出 $\boldsymbol{C}$ 矩阵，从而判定矩阵方程的可解性。

```
>> A=[1 2 3 4; 2 2 1 1; 2 4 6 8; 4 4 2 2];  B=[1;3;2;6];
   C=[A B]; [rank(A), rank(C)]
ans =
     2     2
```

通过检验秩的方法得出矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{C}$ 的秩相同，都等于 2，小于矩阵的阶次 4，由此可以得出结论，原线性代数方程组有无穷多组解。如需求解原代数方程组，可以先求出化零空间 $\boldsymbol{Z}$，并得出满足方程的一个特解 $\boldsymbol{x}_0$。

```
>> Z=null(A,'r')    % 解出规范化的化零空间
Z =
   2.00000000000000   3.00000000000000
  -2.50000000000000  -3.50000000000000
   1.00000000000000                  0
                  0   1.00000000000000
>> x0=pinv(A)*B     % 得出一个特解
x0 =
   0.95419847328244
   0.73282442748092
  -0.07633587786260
  -0.29770992366412
```

这样可以将方程全部的解表示成

$$\boldsymbol{x}=\alpha_1\begin{bmatrix}2\\-2.5\\1\\0\end{bmatrix}+\alpha_2\begin{bmatrix}3\\-3.5\\0\\1\end{bmatrix}+\begin{bmatrix}0.95419847328244\\0.73282442748092\\-0.07633587786260\\-0.29770992366412\end{bmatrix}$$

其中，α_1 和 α_2 为任意常数。为验证得出的解，可以任意取随机变量 α_1 和 α_2 并代入方程，则可以得出误差。从下面的结果看该解还是很精确的。

```
>> a1=randn(1); a2=rand(1);  % 取不同分布的随机数
   x=a1*Z(:,1)+a2*Z(:,2)+x0; norm(A*x-B)
ans =
    4.965068306494546e-015
```

从解析意义上看，直接得出的解是双精度的，不甚精确，可以考虑利用符号运算工具箱更精确地求解原问题，得出方程组的解析解。

```
>> Z=null(sym(A))
Z =
    [    2,    3]
    [ -5/2, -7/2]
    [    1,    0]
    [    0,    1]
```

```
>> x0=sym(pinv(A)*B)
x0 =
   [ 125/131]
   [  96/131]
   [ -10/131]
   [ -39/131]
```

由上面结果可以写出方程的解析解为

$$x=\alpha_1\begin{bmatrix}2\\-5/2\\1\\0\end{bmatrix}+\alpha_2\begin{bmatrix}3\\-7/2\\0\\1\end{bmatrix}+\begin{bmatrix}125/131\\96/131\\-10/131\\-39/131\end{bmatrix}$$

使用前面的验证语句，可以看出，得出的解没有误差。

```
>> a1=randn(1); a2=rand(1);  % 取不同分布的随机数
   x=a1*Z(:,1)+a2*Z(:,2)+x0; norm(double(A*x-B))
ans =
    0
```

更一般地，可以设置 a_1, a_2 为符号变量，可以用下面的语句直接写出通解的表达式。

```
>> syms a1 a2;
   x=a1*Z(:,1)+a2*Z(:,2)+x0
x =
   [      2*a1+3*a2+125/131]
   [ -5/2*a1-7/2*a2+96/131]
   [                a1-10/131]
   [                a2-39/131]
```

亦即 $x_1=2a_1+3a_2+\dfrac{125}{131}$, $x_2=-\dfrac{5}{2}a_1-\dfrac{7}{2}a_2+\dfrac{96}{131}$, $x_3=a_1-\dfrac{10}{131}$, $x_4=a_2-\dfrac{39}{131}$。

③ 若 $\mathrm{rank}(\boldsymbol{A}) \leqslant \mathrm{rank}(\boldsymbol{C})$，则方程组 (4-4-1) 为矛盾方程，这时只能利用 Moore-Penrose 广义逆求解出方程的最小二乘解为 $\boldsymbol{x}$=pinv($\boldsymbol{A}$)*$\boldsymbol{B}$，该解不满足原方程，只能使误差的范数测度 $||\boldsymbol{Ax}-\boldsymbol{B}||$ 取最小值。

【例 4-35】如果 $\boldsymbol{B}$ 矩阵改成 $\boldsymbol{B}=[1,2,3,4]^{\mathrm{T}}$，则通过求解可见

```
>> B=[1:4]'; C=[A B]; [rank(A), rank(C)]
ans =
      2     3
```

这样，$\mathrm{rank}(\boldsymbol{A}) \neq \mathrm{rank}(\boldsymbol{C})$，故原始方程是矛盾方程，不存在任何解。可以使用 pinv() 函数求取 Moore-Penrose 广义逆，从而求出原始方程的最小二乘解为

```
>> x=pinv(A)*B
x =
   0.54656488549618
   0.45496183206107
```

```
   0.04427480916031
  -0.04732824427481
>> norm(A*x-B)
ans =
   0.44721359549996
```

该解不满足原始代数方程组。

如果线性方程为

$$xA = B \tag{4-4-6}$$

则可以对上式两端进行转置处理，得出

$$A^{\mathrm{T}}z = B^{\mathrm{T}} \tag{4-4-7}$$

式中，$z = x^{\mathrm{T}}$，亦即可以得出形为式 (4-4-1) 的新线性方程，套用上述的几种情况则可以求解原始线性方程组。

4.4.2 Lyapunov 方程的计算机求解

4.4.2.1 连续 Lyapunov 方程

连续 Lyapunov 方程可以表示成

$$AX + XA^{\mathrm{T}} = -C \tag{4-4-8}$$

Lyapunov 方程来源于微分方程稳定性理论，其中要求 $-C$ 为对称正定的 $n \times n$ 矩阵，从而可以证明解 X 亦为$n \times n$ 对称矩阵。这类方程直接求解是很困难的，不过有了 MATLAB 这样的计算机数学语言，求解这样的问题就轻而易举了。可以由控制系统工具箱中提供的 lyap() 函数立即得出方程的解，该函数的调用格式为

X=lyap(A,C)

所以若给出 Lyapunov 方程中的 A 和 C，则可以立即获得相应 Lyapunov 方程的数值解。下面将通过例子演示一般 Lyapunov 方程的求解。

【例 4-36】假设式 (4-4-8) 中的 A 和 C 矩阵分别为

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}, \quad C = -\begin{bmatrix} 10 & 5 & 4 \\ 5 & 6 & 7 \\ 4 & 7 & 9 \end{bmatrix}$$

试求解相应的 Lyapunov 方程，并验证解的精度。

【求解】输入了给定的矩阵，可以由下面的 MATLAB 语句求出该方程的解。

```
>> A=[1 2 3;4 5 6; 7 8 0];  C=-[10, 5, 4; 5, 6, 7; 4, 7, 9];
   X=lyap(A,C)
X =
  -3.94444444444445    3.88888888888889    0.38888888888889
```

```
    3.888888888888889  -2.777777777777778   0.222222222222222
    0.388888888888889   0.222222222222222  -0.111111111111111
>> norm(A*X+X*A'+C)
ans =
     2.647423139608994e-014
```

从最后一个语句可见，得出的方程解 $\boldsymbol{X}$ 基本满足原方程，且有较高精度。

4.4.2.2 Lyapunov 方程的解析解

为方便叙述，可以将 Lyapunov 方程的各个矩阵参数表示为

$$\boldsymbol{X}=\begin{bmatrix} x_1 & x_2 & \cdots & x_m \\ x_{m+1} & x_{m+2} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(n-1)m+1} & x_{(n-1)m+2} & \cdots & x_{nm} \end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix} c_1 & c_2 & \cdots & c_m \\ c_{m+1} & c_{m+2} & \cdots & c_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{(n-1)m+1} & c_{(n-1)m+2} & \cdots & c_{nm} \end{bmatrix}$$

利用 Kronecker 乘积的表示方法，可以将 Lyapunov 方程写成

$$(\boldsymbol{A}\otimes\boldsymbol{I}+\boldsymbol{I}\otimes\boldsymbol{A})\boldsymbol{x}=-\boldsymbol{c} \tag{4-4-9}$$

可见，这样的方程有惟一解的条件并不局限于 $-\boldsymbol{C}$ 为对称正定矩阵，形如式 (4-4-8) 的方程只要满足 $(\boldsymbol{A}\otimes\boldsymbol{I}+\boldsymbol{I}\otimes\boldsymbol{A})$ 为非奇异的方阵即可保证惟一解。

【例 4-37】 仍考虑例 4-36 中给出的 Lyapunov 方程，试求出其解析解。

【求解】 由下面的语句可以求出其解析解，将其解代入原方程可以验证这一点。

```
>> A0=sym(kron(A,eye(3))+kron(eye(3),A));
   c=reshape(C',9,1);  x0=-inv(A0)*c; x=reshape(x0,3,3)'
x =
  [ -71/18,    35/9,    7/18]
  [   35/9,   -25/9,     2/9]
  [   7/18,     2/9,    -1/9]
>> norm(double(A*x+x*A'+C))
ans =
     0
```

【例 4-38】 传统 Lyapunov 方程的条件 ($\boldsymbol{C}$ 为实对称正定矩阵) 能否突破?

【求解】 受微分方程稳定性影响，以前的传统观念似乎 Lyapunov 类方程有惟一解的充分必要条件是 $-\boldsymbol{C}$ 矩阵为实对称正定矩阵。事实上，式 (4-4-10) 中给出的线性矩阵方程在不满足该条件的情况下仍有惟一解。例如，例 4-36 中给出的 $\boldsymbol{A}$ 矩阵不变，将 $\boldsymbol{C}$ 矩阵改为复数非对称矩阵

$$\boldsymbol{C}=-\begin{bmatrix} 1+1\mathrm{j} & 3+3\mathrm{j} & 12+10\mathrm{j} \\ 2+5\mathrm{j} & 6 & 11+6\mathrm{j} \\ 5+2\mathrm{j} & 11+\mathrm{j} & 2+12\mathrm{j} \end{bmatrix}$$

用上述方法可以输入 $\boldsymbol{A}$ 和 $\boldsymbol{C}$ 矩阵，可以立即解出满足该方程的复数解为

```
>> A=[1 2 3;4 5 6; 7 8 0];
   C=-[1+1i, 3+3i, 12+10i; 2+5i, 6, 11+6i; 5+2i, 11+1i, 2+12i];
   A0=sym(kron(A,eye(3))+kron(eye(3),A));
   c=reshape(C',9,1);  x0=-inv(A0)*c; x=reshape(x0,3,3)'
x =
   [ -5/102+1457/918*i,   15/17-371/459*i, -61/306+166/459*i]
   [     4/17-626/459*i,  -10/51+160/459*i, 115/153+607/459*i]
   [ -55/306+166/459*i, -26/153-209/459*i, 203/153+719/918*i]
>> norm(double(A*x+x*A+C))
ans =
     0
```

得出的解经验证确实满足原始 Lyapunov 方程。故可以得出结论，如果不考虑 Lyapunov 方程稳定性的物理意义和 Lyapunov 函数为能量的物理原型，完全可以将 Lyapunov 方程进一步扩展成能处理任意 C 矩阵的情形。

4.4.2.3 离散 Lyapunov 方程

离散 Lyapunov 方程的一般表示形式为

$$\boldsymbol{AXA}^{\mathrm{T}}-\boldsymbol{X}+\boldsymbol{Q}=\boldsymbol{0} \tag{4-4-10}$$

该方程可以由 MATLAB 控制系统工具箱的 dlyap() 函数直接求解。该函数的调用格式为

$\boldsymbol{X}$=dlyap($\boldsymbol{A}$,$\boldsymbol{Q}$)

其实，如果 $\boldsymbol{A}$ 矩阵是非奇异矩阵，则等式两端同时右乘 $(\boldsymbol{A}^{\mathrm{T}})^{-1}$，就可以将其变换成连续的 Sylvester 方程，可以用第 4.4.3 节给出的算法求解其解析解。

【例 4-39】求解下面的离散 Lyapunov 方程

$$\begin{bmatrix}8&1&6\\3&5&7\\4&9&2\end{bmatrix}\boldsymbol{X}\begin{bmatrix}8&1&6\\3&5&7\\4&9&2\end{bmatrix}^{\mathrm{T}}-\boldsymbol{X}+\begin{bmatrix}16&4&1\\9&3&1\\4&2&1\end{bmatrix}=\boldsymbol{0}$$

【求解】该方程可以直接用 dlyap() 方程求解出来.

```
>> A=[8,1,6; 3,5,7; 4,9,2]; Q=[16,4,1; 9,3,1; 4,2,1];
   X=dlyap(A,Q)
X =
  -0.16474425474467   0.06914962254340  -0.01678480975888
   0.05284289907609  -0.02978496276628  -0.00615417767726
  -0.10197836412080   0.04495899142813  -0.03054065826545
>> norm(A*X*A'-X+Q)   % 精度验证
ans =
     2.777775429957891e-014
```

4.4.3 Sylvester 方程的计算机求解

Sylvester 方程的一般形式为

$$\boldsymbol{A}\boldsymbol{X}+\boldsymbol{X}\boldsymbol{B}=-\boldsymbol{C} \tag{4-4-11}$$

其中，$\boldsymbol{A}$ 为 $n\times n$ 矩阵，$\boldsymbol{B}$ 为 $m\times m$ 矩阵，$\boldsymbol{C}$ 和 $\boldsymbol{X}$ 均为 $n\times m$ 矩阵。该方程又称为广义的 Lyapunov 方程，式中 $\boldsymbol{A}$ 为 $n\times n$ 矩阵，$\boldsymbol{B}$ 为 $m\times m$ 矩阵。仍可以利用 MATLAB 控制系统工具箱中的 `lyap()` 函数直接求解该方程。该函数的一般调用格式为

X=lyap(**A**, **B**, **C**)

该函数采用的是 Schur 分解的数值解法求解方程。如果想得到解析解，类似于前述的一般 Lyapunov 方程，可以采用 Kronecker 乘积的形式将原始方程进行如下变换：

$$(\boldsymbol{A}\otimes\boldsymbol{I}_m+\boldsymbol{I}_n\otimes\boldsymbol{B}^{\mathrm{T}})\boldsymbol{x}=\boldsymbol{c} \tag{4-4-12}$$

如果 $(\boldsymbol{A}\otimes\boldsymbol{I}_m+\boldsymbol{I}_n\otimes\boldsymbol{B}^{\mathrm{T}})$ 矩阵为非奇异矩阵，则 Sylvester 方程有惟一解。

综合上述的算法，可以编写出 Sylvester 型方程的解析解求解程序 lyap.m，将其置于 `@sym` 目录下，以后再求解时只需将 $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}$ 矩阵之一设置成符号变量，就可以直接调用该函数了。这样改写的函数清单为

```
function X=lyap(A,B,C)  % 注意应该置于 @sym 目录下
if nargin==2, C=B; B=A'; end
[nr,nc]=size(C);
A0=kron(A,eye(nc))+kron(eye(nr),B');
try
   C1=C'; x0=-inv(A0)*C1(:);
   X=reshape(x0,nc,nr)';
catch, error('singular matrix found.'), end
```

考虑式 (4-4-10) 中给出的离散 Lyapunov 方程，两端同时右乘 $(\boldsymbol{A}^{\mathrm{T}})^{-1}$，则原来的离散 Lyapunov 方程可以变换成

$$\boldsymbol{A}\boldsymbol{X}+\boldsymbol{X}[-(\boldsymbol{A}^{\mathrm{T}})^{-1}]=-\boldsymbol{Q}(\boldsymbol{A}^{\mathrm{T}})^{-1}$$

故令 $\boldsymbol{B}=-(\boldsymbol{A}^{\mathrm{T}})^{-1}$，$\boldsymbol{C}=\boldsymbol{Q}(\boldsymbol{A}^{\mathrm{T}})^{-1}$，则可以将其变换成式 (4-4-11) 所示的 Sylvester 方程，故也可以通过新的 `lyap()` 函数求解该方程。该函数的具体调用格式为

X=lyap(sym(**A**),**C**)	连续 Lyapunov 方程
X=lyap(sym(**A**),-inv(**A**'),**Q***inv(**A**'))	离散 Lyapunov 方程
X=lyap(sym(**A**),**B**,**C**)	Sylvester 方程

【例 4-40】求解下面的 Sylvester 方程。

$$\begin{bmatrix}8&1&6\\3&5&7\\4&9&2\end{bmatrix}\boldsymbol{X}+\boldsymbol{X}\begin{bmatrix}16&4&1\\9&3&1\\4&2&1\end{bmatrix}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix}$$

【求解】 调用 lyap() 函数可以立即得出原方程的数值解为

```
>> A=[8,1,6; 3,5,7; 4,9,2]; B=[16,4,1; 9,3,1; 4,2,1];
   C=-[1,2,3; 4,5,6; 7,8,0];  X=lyap(A,B,C)
X =
   0.07487187370025   0.08991343376264  -0.43292000329628
   0.00807164473631   0.48144176804999  -0.21603391285553
   0.01957708262984   0.18264382872543   1.15792143917653
>> norm(A*X+X*B+C)
ans =
    4.531536518319095e-015
```

经检验可见该解精度较高。如果想获得原方程的解析解，则可以使用下面的语句，并可验证得出的解确实满足原始方程。

```
>> x=lyap(sym(A),B,C)
x =
   [    1349214/18020305,       648107/7208122, -15602701/36040610]
   [     290907/36040610,      3470291/7208122,  -3892997/18020305]
   [       70557/3604061,      1316519/7208122,     8346439/7208122]
>> norm(double(A*x+x*B+C))
ans =
     0
```

【例 4-41】 重新考虑例 4-39 中给出的离散 Lyapunov 方程，试求取其解析解。

【求解】 该方程可以通过下面的语句求解出解析解。

```
>> A=[8,1,6; 3,5,7; 4,9,2]; Q=[16,4,1; 9,3,1; 4,2,1];
   x=lyap(sym(A),-inv(A'),Q*inv(A'))
x =
   [ -22912341/139078240,  48086039/695391200, -11672009/695391200]
   [  36746487/695391200, -20712201/695391200,   -4279561/695391200]
   [ -70914857/695391200,  31264087/695391200,   -4247541/139078240]
>> norm(double(A*x*A'-x+Q))  % 可以证明这样的解没有误差
ans =
     0
```

【例 4-42】 求解下面的 Sylvester 方程。

$$\boldsymbol{A}=\begin{bmatrix}8&1&6\\3&5&7\\4&9&2\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}2&3\\4&5\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}1&2\\3&4\\5&6\end{bmatrix}$$

【求解】 Sylvester 方程能解决的问题中并未要求 $\boldsymbol{C}$ 矩阵为方阵，利用上面的语句仍然能求出此方程的解析解，这里还可以尝试上面编写的 Lyapunov 方程解析解求解的新函数 lyap()，可以直接求解上述的方程。

```
>> A=[8,1,6; 3,5,7; 4,9,2]; B=[2,3; 4,5]; C=-[1,2; 3,4; 5,6];
   X=lyap(sym(A),B,C)
X =
    [ -2853/14186, -11441/56744]
    [  -557/14186,  -8817/56744]
    [  9119/14186,  50879/56744]
>> norm(double(A*X+X*B+C))   % 经检验没有误差
ans =
     0
```

4.4.4 Riccati 方程的计算机求解

Riccati 方程是一类很著名的二次型矩阵方程式，其一般形式为

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{X}\boldsymbol{A}-\boldsymbol{X}\boldsymbol{B}\boldsymbol{X}+\boldsymbol{C}=\boldsymbol{0} \tag{4-4-13}$$

由于含有未知矩阵 $\boldsymbol{X}$ 的二次项，所以 Riccati 方程的求解数学上要比 Lyapunov 方程更难。MATLAB 的控制系统工具箱中提供了现成函数 are()，可以直接求解式 (4-4-13) 中给出的方程，该函数的具体调用格式为

X=are(A,B,C)

【例 4-43】 考虑式 (4-4-13) 中给出的 Riccati 方程，其中

$$\boldsymbol{A}=\begin{bmatrix}-2&1&-3\\-1&0&-2\\0&-1&-2\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}2&2&-2\\-1&5&-2\\-1&1&2\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}5&-4&4\\1&0&4\\1&-1&5\end{bmatrix}$$

试求出该方程的数值解，并验证解的正确性。

【求解】 可以用下面的语句直接求解该方程。

```
>> A=[-2,1,-3; -1,0,-2; 0,-1,-2]; B=[2,2,-2; -1 5 -2; -1 1 2];
   C=[5 -4 4; 1 0 4; 1 -1 5]; X=are(A,B,C)
X =
    0.98739490849791  -0.79832769688830   0.41886899662564
    0.57740564955473  -0.13079233649093   0.57754776836148
   -0.28404500018051  -0.07303697833280   0.69241148830571
>> norm(double(A'*X+X*A-X*B*X+C))   % 验证结果可见，结果很精确
ans =
    1.860508358348165e-014
```

4.5　非线性运算与矩阵函数求值

4.5.1　面向矩阵元素的非线性运算

MATLAB 提供了大量函数，允许用户对矩阵进行处理，前面介绍的主要是矩阵的线性变换，本节将介绍如何对矩阵进行非线性运算。

事实上，MATLAB 提供了两类函数，其中一类是对矩阵的各个元素进行单独运算的，而另一类是对整个矩阵进行运算的。前面曾经用到了 `sin()` 函数，该函数属于第一类，是对矩阵的各个元素单独运算的，而不是对整个矩阵进行运算的。这类常用的 MATLAB 函数在表 4-2 中列出来，它们的调用方法是很显然的，其标准调用格式为

```
B = 函数名(A); 例如 B=sin(A);
```

表 4-2　面向矩阵元素的非线性函数表

函数名	意　义	函数名	意　义
abs()	求模（绝对值）函数	asin(), acos(), atan()	反正弦、余弦、正切函数
sqrt()	求平方根函数	log(), log10()	自然和常用对数
exp()	指数函数	real(), imag(), conj()	求实虚部及共轭复数
sin(), cos(), tan()	正弦、余弦、正切函数	round(), floor(), ceil()	取整数函数

【例 4-44】考虑例 4-7 中给出的 A 矩阵，调用其中的一些函数，其结果在下面给出。

```
>> A=[16,2,3,13; 5,11,10,8; 9,7,6,12; 4,14,15,1]; exp(A)
ans =
   1.0e+006 *
  8.88611052050787   0.00000738905610   0.00002008553692   0.44241339200892
  0.00014841315910   0.05987414171520   0.02202646579481   0.00298095798704
  0.00810308392758   0.00109663315843   0.00040342879349   0.16275479141900
  0.00005459815003   1.20260428416478   3.26901737247211   0.00000271828183
>> sin(A)
ans =
 -0.28790331666507   0.90929742682568   0.14112000805987   0.42016703682664
 -0.95892427466314  -0.99999020655070  -0.54402111088937   0.98935824662338
  0.41211848524176   0.65698659871879  -0.27941549819893  -0.53657291800043
 -0.75680249530793   0.99060735569487   0.65028784015712   0.84147098480790
```

4.5.2　矩阵函数求值

4.5.2.1　矩阵指数的运算

除了对矩阵的单个元素进行单独计算以外，一般还常常要求对整个矩阵做这样的非线性运算。例如，想求出一个矩阵的 e 指数，就需要特殊的算法来完成了。文献 [32]

中叙述了求解矩阵指数的 19 种不同方法，每一种方法都有自己的特点及适用范围。在 MATLAB 中提供了 4 个求取矩阵指数的函数，分别 expm()，expm1()，expm2() 和 expm3()。其中，expm() 为内在函数，其调用格式为

E=expm(A)

该函数采用 Padé 近似技术来求取矩阵的指数。而 expm1() 函数是 expm() 函数的 M-函数实现。函数 expm2() 采用 Taylor 级数展开方法来求取矩阵的指数，该方法比较直观，直接对矩阵指数作幂级数展开。

$$\mathrm{e}^{\boldsymbol{A}}=\sum_{i=0}^{\infty}\frac{1}{i!}\boldsymbol{A}^i=\boldsymbol{I}+\boldsymbol{A}+\frac{1}{2}\boldsymbol{A}^2+\frac{1}{3!}\boldsymbol{A}^3+\cdots+\frac{1}{m!}\boldsymbol{A}^m+\cdots \tag{4-5-1}$$

可以看出，这样的运算可以由 while 循环结构来编程，当幂级数累加项的范数满足误差要求时退出循环即可。

expm3() 采用特征值特征向量的方法求出矩阵的指数矩阵。其数学原理为：首先求出矩阵 $\boldsymbol{A}$ 的特征值 $\boldsymbol{D}=\mathrm{diag}(\gamma_1,\gamma_2,\cdots,\gamma_n)$ 及相应的特征向量矩阵 $\boldsymbol{V}$，然后对该对角矩阵求取矩阵指数，即对每个对角矩阵元素求指数，这时原矩阵 $\boldsymbol{A}$ 的指数矩阵为

$$\mathrm{e}^{\boldsymbol{A}}=\boldsymbol{V}\begin{bmatrix}\mathrm{e}^{\gamma_1} & & \\ & \ddots & \\ & & \mathrm{e}^{\gamma_n}\end{bmatrix}\boldsymbol{V}^{-1} \tag{4-5-2}$$

这种方法看似简单，但有很大的局限性，它一般要求原矩阵没有重根，否则往往得出的特征向量矩阵趋于奇异，因而可能得出错误的结果。对这样的问题将引入广义特征向量的概念。MATLAB 7.0 版本中取消了 expm2() 和 expm3() 函数，避免了这两个不能保证可靠性的函数使用。下面将给出演示矩阵指数求取的例子。

【例 4-45】考虑下面给出的矩阵

$$\boldsymbol{A}=\begin{bmatrix}-2 & 1 & 0 & & \\ 0 & -2 & 1 & & \\ 0 & 0 & -2 & & \\ & & & -5 & 1\\ & & & 0 & -5\end{bmatrix}$$

试求出该矩阵的指数和对数，即 $\mathrm{e}^{\boldsymbol{A}}$ 和 $\ln\boldsymbol{A}$。

【求解】如果对此矩阵进行指数运算和对数运算，则可以获得以下的结果：

```
>> A=[[-2 1 0; 0 -2 1; 0 0 -2], zeros(3,2); zeros(2,3) [-5 1; 0 -5]];
   expm(A)   % 数值解求解
ans =
      0.1353    0.1353    0.0677         0         0
           0    0.1353    0.1353         0         0
           0         0    0.1353         0         0
           0         0         0    0.0067    0.0067
```

```
         0         0         0         0    0.0067
>> logm(ans)   % 数值解求对数
ans =
   -2.0000    1.0000    0.0000   -0.0000    0.0000
    0.0000   -2.0000    1.0000    0.0000    0.0000
   -0.0000   -0.0000   -2.0000    0.0000    0.0000
    0.0000    0.0000   -0.0000   -5.0000    1.0000
    0.0000    0.0000   -0.0000    0.0000   -5.0000
>> norm(ans-A)
ans =
    2.6762e-014
```

对得出的指数结果进行对数运算可以按相当高的精度还原原始矩阵，从而表明，矩阵对数运算还是很精确的。

原始问题还可以调用解析解函数 expm()，直接求解 e^{At}。注意，这里包含了变量 t，所以这是数值算法无法解出的。

```
>> syms t; expm(A*t)
ans =
  [ exp(-2*t), t*exp(-2*t), 1/2*t^2*exp(-2*t),         0,           0]
  [         0,  exp(-2*t),        t*exp(-2*t),         0,           0]
  [         0,          0,          exp(-2*t),         0,           0]
  [         0,          0,                  0, exp(-5*t), t*exp(-5*t)]
  [         0,          0,                  0,         0,   exp(-5*t)]
```

同样的问题利用 MATLAB 下的 expm3() 函数求数值解，将出现问题。

```
>> format short; expm3(A)  % 显示精度降低，利于排版
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.972152e-031.
> In C:\MATLAB6p5p1\toolbox\matlab\demos\expm3.m at line 13
ans =
    0.1353         0         0         0         0
         0    0.1353         0         0         0
         0         0    0.1353         0         0
         0         0         0    0.0067         0
         0         0         0         0    0.0067
```

究其原因，会发现该算法由于使用了特征向量矩阵，故对有重根矩阵的运算因特征向量矩阵接近奇异而出现数值错误，故这样的问题不适合用该算法求解。

由于该矩阵已经由 Jordan 矩阵形式给出，所以可以直接写出

$$e^{\boldsymbol{A}t}=\begin{bmatrix} e^{-2t} & te^{-2t} & t^2e^{-2t}/2 & 0 & 0\\ 0 & e^{-2t} & te^{-2t} & 0 & 0\\ 0 & 0 & e^{-2t} & 0 & 0\\ 0 & 0 & 0 & e^{-5t} & te^{-5t}\\ 0 & 0 & 0 & 0 & e^{-5t}\end{bmatrix}$$

【例 4-46】 已知矩阵 $\boldsymbol{A}$，试求出 $e^{\boldsymbol{A}t}$。

$$\boldsymbol{A}=\begin{bmatrix} -3 & -1 & -1\\ 0 & -3 & -1\\ 1 & 2 & 0\end{bmatrix}$$

【求解】 如果 Jordan 标准型不那么明显，则不能采用直接写出的方法求解 $e^{\boldsymbol{A}t}$，而应该采用广义特征向量矩阵的方式进行变换。现在考虑

```
>> syms t; A=[-3,-1,-1; 0,-3,-1; 1,2,0]; simple(expm(A*t))
ans =
[     -exp(-2*t)*(-1+t),                    -t*exp(-2*t),                 -t*exp(-2*t)]
[   -1/2*t^2*exp(-2*t),-1/2*exp(-2*t)*(-2+2*t+t^2),   -1/2*t*exp(-2*t)*(t+2)]
[1/2*t*exp(-2*t)*(t+2),       1/2*t*exp(-2*t)*(t+4),1/2*exp(-2*t)*(2+t^2+4*t)]
```

下面演示基于 Jordan 矩阵变换的 $e^{\boldsymbol{A}t}$ 矩阵处理方法。

```
>> [V,J]=jordan(A)    % Jordan 矩阵变换
V =
     0    -1     1
    -1     0     0
     1     1     0
J =
    -2     1     0
     0    -2     1
     0     0    -2
```

可以得出 Jordan 矩阵 $\boldsymbol{J}$ 和广义特征向量矩阵 $\boldsymbol{V}$。由 Jordan 矩阵可以写出 $e^{\boldsymbol{J}t}$ 的表达式为

```
>> J1=[exp(-2*t), t*exp(-2*t), 1/2*t^2*exp(-2*t);
                  0,    exp(-2*t),        t*exp(-2*t);
                  0,            0,          exp(-2*t)];
```

这样，原矩阵的指数矩阵可以由下面的指令求出，其结果与直接求解的结果是完全一致的。

```
>> A1=simple(V*J1*inv(V))
A1 =
[     -exp(-2*t)*(-1+t),                    -t*exp(-2*t),                 -t*exp(-2*t)]
[   -1/2*t^2*exp(-2*t),-1/2*exp(-2*t)*(-2+2*t+t^2),   -1/2*t*exp(-2*t)*(t+2)]
[1/2*t*exp(-2*t)*(t+2),       1/2*t*exp(-2*t)*(t+4),1/2*exp(-2*t)*(2+t^2+4*t)]
```

其实，用这样的方法求解矩阵指数不是此例子的目的，因为用符号运算工具箱中的 expm() 函数可以立即得出所需的结果。后面将通过例子演示其他函数，如正弦等函数如何用 Jordan 矩阵的方法求解。

4.5.2.2 矩阵的三角函数运算

MATLAB 下没有对矩阵进行三角函数运算的现成函数，求解其数值解可以通过 funm() 函数。该函数的目的是求出矩阵的任意函数，其调用方法为

A_1=funm(A,'函数名')

其中，函数名应该由单引号括起来。例如，若想求出矩阵 A 的正弦矩阵，则可以使用命令 B=funm(A,'sin')。值得指出的是，这里给出的矩阵函数运算是基于矩阵特征值特征向量而完成的，类似于 expm3() 的效果，故在一些特殊但很常见的场合 (如有重根矩阵) 下仍将出现错误。

【例 4-47】 重新考虑例 4-45 中给出的矩阵，如果想对其中的 A 矩阵进行正弦运算，则将得出如下的错误结论。

```
>> A=[[-2 1 0; 0 -2 1; 0 0 -2], zeros(3,2); zeros(2,3) [-5 1; 0 -5]];
   funm(A,'sin')
Warning: Result from FUNM may be inaccurate. esterr = 1.
(Type "warning off MATLAB:funm:PossibleInaccuracy" to suppress this warning.)
> In C:\MATLAB6p5p1\toolbox\matlab\matfun\funm.m at line 76
ans =
   -0.9093         0         0         0         0
         0   -0.9093         0         0         0
         0         0   -0.9093         0         0
         0         0         0    0.9589         0
         0         0         0         0    0.9589
```

MATLAB 7.0 修改了 funm() 中的算法，改用 Taylor 幂级数展开的方式计算矩阵函数，可以得出正确的结果，但这样要求函数有 Taylor 幂级数展开表达式。

事实上，矩阵的非线性函数运算可以通过幂级数的方法简单地求出。例如，正弦函数可以由下面的幂级数展开式求出。

$$\sin \boldsymbol{A}=\sum_{i=0}^{\infty}(-1)^i\frac{\boldsymbol{A}^{2i+1}}{(2i+1)!}=\boldsymbol{A}-\frac{1}{3!}\boldsymbol{A}^3+\frac{1}{5!}\boldsymbol{A}^5+\cdots \tag{4-5-3}$$

可以用 MATLAB 实现正弦函数幂级数的展开。

```
function E=sinm1(A)
E = zeros(size(A)); F = A; k = 1;
while norm(E+F-E,1) > 0
   E = E + F;  F = -A^2*F/((k+2)*(k+1)); k = k+2;
end
```

【例 4-48】由上面的程序可以看出，看起来比较复杂的矩阵正弦函数的幂级数展开运算可以由几条 MATLAB 语句容易地编写出来。利用该函数可以容易地求出原矩阵 $\boldsymbol{A}$ 的正弦矩阵为

```
>> E=sinm1(A)
E =
   -0.9093   -0.4161    0.4546         0         0
         0   -0.9093   -0.4161         0         0
         0         0   -0.9093         0         0
         0         0         0    0.9589    0.2837
         0         0         0         0    0.9589
```

可以测出，该函数一共进行了 39 次叠加运算。对上面的结果矩阵再进行反正弦运算，不难得出这样的结论，这种运算可以还原出原来的矩阵 $\boldsymbol{A}$，而这样的结果光靠 MATLAB 提供的 funm() 函数是不可能得出的。所以在使用 funm() 函数时应该格外注意，如果确实不能得出正确的结果，则建议采用幂级数的方法编写程序来直接求出。

再考虑矩阵三角函数的解析解求解方法。先考虑标量三角函数的运算公式，根据著名的 Euler 公式 $\mathrm{e}^{\mathrm{j}a}=\cos a+\mathrm{j}\sin a$ 与 $\mathrm{e}^{-\mathrm{j}a}=\cos a-\mathrm{j}\sin a$ 可以立即推导出

$$\sin a=\frac{1}{\mathrm{j}2}(\mathrm{e}^{\mathrm{j}a}-\mathrm{e}^{-\mathrm{j}a}),\quad \cos a=\frac{1}{2}(\mathrm{e}^{\mathrm{j}a}+\mathrm{e}^{-\mathrm{j}a}) \tag{4-5-4}$$

此公式可以直接用于 a 为矩阵的形式。下面通过例子演示一般矩阵的正弦和余弦函数的解析解运算。

【例 4-49】仍考虑例 4-45 中给出的矩阵，试求解 $\sin\boldsymbol{A}$。

【求解】可以利用现成的 expm() 函数求出矩阵的正弦函数。

```
>> A=[[-2 1 0; 0 -2 1; 0 0 -2], zeros(3,2); zeros(2,3) [-5 1; 0 -5]];
   j=sqrt(-1);
   A1=(expm(A*j)-expm(-A*j))/(2*j),
A1 =
   -0.9093   -0.4161    0.4546         0         0
         0   -0.9093   -0.4161         0         0
         0         0   -0.9093         0         0
         0         0         0    0.9589    0.2837
         0         0         0         0    0.9589
```

可见，精确的解与例 4-48 完全一致，证明该解是正确的。

【例 4-50】假设给出如下的矩阵

$$\boldsymbol{A}=\begin{bmatrix}-7&2&0&-1\\1&-4&2&1\\2&-1&-6&-1\\-1&-1&0&-4\end{bmatrix}$$

已知该矩阵有重特征根，试求出该矩阵的正弦函数 $\sin\boldsymbol{A}t$ 和余弦函数 $\cos\boldsymbol{A}t$。

【求解】根据式 (4-5-4) 可以由下面的语句求解矩阵的正弦和余弦函数

```
>> A=[-7,2,0,-1; 1,-4,2,1; 2,-1,-6,-1; -1,-1,0,-4];
   syms t; j=sym(sqrt(-1));
   A1=simple((expm(A*j*t)-expm(-A*j*t))/(2*j)),
   A2=simple((expm(A*j*t)+expm(-A*j*t))/2)
```

由于结果过于冗长，这里只列出其 LaTeX 表示的结果如下：

$$\sin \boldsymbol{A}t = \begin{bmatrix} -2/9\sin 3t + (t^2-7/9)\sin 6t - 5/3t\cos 6t & -1/3\sin 3t + 1/3\sin 6t + t\cos 6t \\ -2/9\sin 3t + (t^2+2/9)\sin 6t + 1/3t\cos 6t & -1/3\sin 3t - 2/3\sin 6t + t\cos 6t \\ -2/9\sin 3t + (-2t^2+2/9)\sin 6t + 4/3t\cos 6t & -1/3\sin 3t + 1/3\sin 6t - 2t\cos 6t \\ 4/9\sin 3t + (t^2-4/9)\sin 6t + 1/3t\cos 6t & 2/3\sin 3t - 2/3\sin 6t + t\cos 6t \end{bmatrix}$$

$$\begin{bmatrix} -2/9\sin 3t + (2/9+t^2)\sin 6t - 2/3t\cos 6t & 1/9\sin 3t + (-1/9+t^2)\sin 6t - 2/3t\cos 6t \\ -2/9\sin 3t + (2/9+t^2)\sin 6t + 4/3t\cos 6t & 1/9\sin 3t + (-1/9+t^2)\sin 6t + 4/3t\cos 6t \\ -2/9\sin 3t - (7/9+2t^2)\sin 6t - 2/3t\cos 6t & 1/9\sin 3t - (1/9+2t^2)\sin 6t - 2/3t\cos 6t \\ 4/9\sin 3t + (-4/9+t^2)\sin 6t + 4/3t\cos 6t & -2/9\sin 3t + (-7/9+t^2)\sin 6t + 4/3t\cos 6t \end{bmatrix}$$

$$\cos \boldsymbol{A}t = \begin{bmatrix} 2/9\cos 3t + (-t^2+7/9)\cos 6t - 5/3t\sin 6t & 1/3\cos 3t - 1/3\cos 6t + t\sin 6t \\ 2/9\cos 3t - (t^2+2/9)\cos(6*t) + 1/3t\sin 6t & 1/3\cos 3t + 2/3\cos 6t + t\sin 6t \\ 2/9\cos 3t + (2t^2-2/9)\cos 6t + 4/3t\sin 6t & 1/3\cos 3t - 1/3\cos 6t - 2t\sin 6t \\ -4/9\cos 3t + (-t^2+4/9)\cos 6t + 1/3t\sin 6t & -2/3\cos 3t + 2/3\cos 6t + t\sin 6t \end{bmatrix}$$

$$\begin{bmatrix} 2/9\cos 3t - (2/9+t^2)\cos 6t - 2/3t\sin 6t & -1/9\cos 3t + (1/9-t^2)\cos 6t - 2/3t\sin 6t \\ 2/9\cos 3t - (2/9+t^2)\cos 6t + 4/3t\sin 6t & -1/9\cos 3t + (1/9-t^2)\cos 6t + 4/3t\sin 6t \\ 2/9\cos 3t + (7/9+2t^2)\cos 6t - 2/3t\sin 6t & -1/9\cos 3t + (1/9+2t^2)\cos 6t - 2/3t\sin 6t \\ -4/9\cos 3t + (4/9-t^2)\cos 6t + 4/3t\sin 6t & 2/9\cos 3t + (7/9-t^2)\cos 6t + 4/3t\sin 6t \end{bmatrix}$$

4.5.2.3 一般矩阵函数的运算

除了对整个矩阵求取矩阵指数之外，MATLAB 还允许求取矩阵的其他非线性函数，其中常用的函数还有 `logm()` (矩阵求对数)、`sqrtm()` (矩阵求平方根) 和 `funm()` (矩阵求任意函数) 等。可以看出，这里的函数名很有特点，每个函数名在标准函数名的后面加了一个后缀 m，表示对矩阵而不是对矩阵元素进行运算。

遗憾的是，现有的 `funm()` 函数是基于特征值和特征向量矩阵的，所以矩阵有重根时，由于特征向量矩阵奇异，故得出的结果是不可靠的，甚至是错误的。这里将介绍基于 Jordan 矩阵的矩阵函数求解方法[19]。

首先可以将 m_i 阶 Jordan 块 $\boldsymbol{J}_i$ 写成 $\boldsymbol{J}_i = \lambda_i\boldsymbol{I} + \boldsymbol{H}_{m_i}$，其中，$\lambda_i$ 为 Jordan 矩阵的重特征值，$\boldsymbol{H}_{m_i}$ 为幂零矩阵，即 $k \geqslant m_i$ 时 $\boldsymbol{H}_{m_i}^k \equiv \boldsymbol{0}$。这样可以证明，矩阵函数 $\psi(\boldsymbol{J}_i)$ 可以由下式求出。

$$\psi(\boldsymbol{J}_i) = \psi(\lambda_i)\boldsymbol{I}_{m_i} + \psi'(\lambda_i)\boldsymbol{H}_{m_i} + \cdots + \frac{\psi^{(m_i-1)}(\lambda_i)}{(m_i-1)!}\boldsymbol{H}_{m_i}^{m_i-1} \tag{4-5-5}$$

如果通过 Jordan 矩阵分解的方法可以将任意矩阵 $\boldsymbol{A}$ 分解成

$$\boldsymbol{A} = \boldsymbol{V}\begin{bmatrix} \boldsymbol{J}_1 & & & \\ & \boldsymbol{J}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{J}_m \end{bmatrix}\boldsymbol{V}^{-1} \tag{4-5-6}$$

这样，该矩阵的任意函数 $\psi(\boldsymbol{A})$ 可以最终如下求出。如果通过 Jordan 矩阵分解的方法可以将任意矩阵 $\boldsymbol{A}$ 分解成

$$\psi(\boldsymbol{A})=\boldsymbol{V}\begin{bmatrix}\psi(\boldsymbol{J}_1) & & & \\ & \psi(\boldsymbol{J}_2) & & \\ & & \ddots & \\ & & & \psi(\boldsymbol{J}_m)\end{bmatrix}\boldsymbol{V}^{-1} \tag{4-5-7}$$

根据上面的算法可以立即编写出新的 funm() 函数，应该置于 @sym 目录下，可以推导任意矩阵函数的解析解。该函数的清单为

```
function F=funm(A,fun,x)
[V,J]=jordan(A); v1=[0,diag(J,1)']; v2=[find(v1==0), length(v1)+1];
for i=1:length(v2)-1
    v_lambda(i)=J(v2(i),v2(i)); v_n(i)=v2(i+1)-v2(i);
end
m=length(v_lambda); F=sym([]);
for i=1:m
    J1=J(v2(i):v2(i)+v_n(i)-1,v2(i):v2(i)+v_n(i)-1);
    fJ=funJ(J1,fun,x); F=diagm(F,fJ);
end
F=V*F*inv(V);
function fJ=funJ(J,fun,x)
lam=J(1,1); f1=fun; fJ=subs(fun,x,lam)*eye(size(J));
H=diag(diag(J,1),1); H1=H;
for i=2:length(J)
   f1=diff(f1,x); a1=subs(f1,x,lam); fJ=fJ+a1*H1; H1=H1*H/i;
end
```

该函数的调用格式为

$\boldsymbol{A}_1$=funm($\boldsymbol{A}$, funx, x)

其中，x 为符号型自变量，funx 为 x 的函数表示。例如，若想求出 $\mathrm{e}^{\boldsymbol{A}}$，则可以将 funx 填写成 exp(x)。其实，funx 参数可以描述任意复杂的函数，如 exp(x*t) 表示求取 $\mathrm{e}^{\boldsymbol{A}t}$，其中 t 也应该事先设置成符号变量。另外，该函数还可以表示成 exp(x*cos(x*t)) 型的复合函数，表示需要求取 $\psi(\boldsymbol{A})=\mathrm{e}^{\boldsymbol{A}\cos(\boldsymbol{A}t)}$。

【例 4-51】已知给定矩阵 $\boldsymbol{A}$，试求出矩阵函数 $\psi(\boldsymbol{A})=\mathrm{e}^{\boldsymbol{A}\cos(\boldsymbol{A}t)}$。

$$\boldsymbol{A}=\begin{bmatrix}-7 & 2 & 0 & -1\\ 1 & -4 & 2 & 1\\ 2 & -1 & -6 & -1\\ -1 & -1 & 0 & -4\end{bmatrix}$$

【求解】可以用下面的语句将其输入到 MATLAB 环境中。

```
>> A=[-7,2,0,-1; 1,-4,2,1; 2,-1,-6,-1; -1,-1,0,-4];
```

如果想求出 $\psi(\boldsymbol{A})=\mathrm{e}^{\boldsymbol{A}\cos(\boldsymbol{A}t)}$，分析题意，可以用下面的语句

```
>> syms x t; A1=funm(sym(A),exp(x*cos(x*t)),x)
```

得出的结果是很冗长的，根本无法显示全部内容，这里只给出其中一项为

$$\psi_{1,1}(\boldsymbol{A})=2/9\mathrm{e}^{-3\cos 3t}+(2t\sin 6t+6t^2\cos 6t)\mathrm{e}^{-6\cos 6t}+(\cos 6t-6t\sin 6t)^2\mathrm{e}^{-6\cos 6t}-5/3(\cos 6t-6t\sin 6t)\mathrm{e}^{-6\cos 6t}+7/9\mathrm{e}^{-6\cos 6t}$$

可见，这样得出的 $\psi_{1,1}(t)$ 有很多项均是 $\mathrm{e}^{-6\cos 6t}$ 的系数项，故可以通过合并同类项的化简方法手动给出下面的命令：

```
>> collect(A1(1,1),exp(-6*cos(6*t)))
```

则可以得出如下的化简结果：

$$\psi_{1,1}(\boldsymbol{A})=\left[12t\sin 6t+6t^2\cos 6t+(\cos 6t-6t\sin 6t)^2-\frac{5}{3}\cos 6t+\frac{7}{9}\right]\mathrm{e}^{-6\cos 6t}+\frac{2}{9}\mathrm{e}^{-3\cos 3t}$$

进一步地，若令 $t=1$，则可以求出 $\mathrm{e}^{\boldsymbol{A}\cos\boldsymbol{A}}$ 的精确数值解为

```
>> subs(A1,t,1)
ans =
   4.35831539990954   6.50441091578582   4.36346744190690  -2.13264353819237
   4.37176737759345   6.50755880947501   4.38006731328000  -2.11604366681927
   4.26528707638475   6.47951110872617   4.25183509870084  -2.24742377508763
  -8.62045458260509 -12.98392202451199  -8.61215464691854   4.38321520696919
```

4.6 本章要点简介

- 线性代数问题的有关 MATLAB 函数如下表给出。

函数名	函 数 功 能	工具箱	本书页码
ones()	生成幺矩阵，即矩阵全部元素均为 1 的矩阵	MATLAB	85
zeros()	生成零矩阵	MATLAB	85
rand()	生成 [0, 1] 区间均匀分布的随机数矩阵	MATLAB	85
randn()	生成标准正态分布 $N(0,1)$ 的随机数矩阵	MATLAB	85
diag()	生成对角矩阵或由一般矩阵提取对角线元素的函数	MATLAB	86
diagm()	生成对角块矩阵	自编	87
hankel()	生成 Hankel 矩阵，编写了同名符号函数，见 90 页	MATLAB	87
vander()	生成 Vandermonde 矩阵，编写了同名符号函数，见 90 页	MATLAB	89
hilb()	生成 Hilbert 矩阵	MATLAB	88
invhilb()	生成 Hilbert 逆矩阵	MATLAB	88
compan()	由多项式构造伴随矩阵，编写了同名符号函数，见 90 页	MATLAB	89

续表

函数名	函 数 功 能	工具箱	本书页码
sym	将已知矩阵转换成符号矩阵	符号运算	90
det()	求矩阵的行列式，同样支持符号运算	MATLAB	91
trace()	求矩阵的迹，同样支持符号运算	MATLAB	92
rank()	求矩阵的秩，同样支持符号运算	MATLAB	92
norm()	求矩阵的各种范数，不支持符号运算	MATLAB	94
poly()	求矩阵的特征多项式，由于算法原因建议采用新同名函数，见 95 页	MATLAB	95
polyvalm()	矩阵的多项式运算，同样支持符号运算	MATLAB	96
polyval()	矩阵的多项式点运算，同样支持符号运算	MATLAB	96
poly2sym()	数值向量转换成符号多项式	符号运算	97
sym2poly()	符号多项式转换成数值向量	符号运算	97
inv()	矩阵求逆，同样支持符号运算	MATLAB	98
pinv()	矩阵的 Moore-Penrose 广义逆，不支持符号运算	MATLAB	101
eig()	求取矩阵的特征值、特征向量或广义特征值，同样适合于符号运算	MATLAB	104,106
orth()	矩阵的正交基计算，不支持符号运算	MATLAB	107
lu()	矩阵的 LU 分解，不支持符号运算	MATLAB	109
chol()	对称矩阵的 Cholesky 分解，不支持符号运算	MATLAB	111,112
jordan()	符号矩阵的 Jordan 矩阵转换	符号运算	114
svd()	矩阵的奇异值分解，支持符号运算	MATLAB	116
null()	矩阵的化零空间或基础解系计算，支持符号运算	MATLAB	119
lyap()	求解连续 Lyapunov 方程、Sylvester 方程的数值解	控制系统	122
dlyap()	求解离散 Lyapunov 方程的数值解	控制系统	124
lyap()	求解连续、离散 Lyapunov 方程、Sylvester 方程的解析解	自编	124
are()	求解 Riccati 方程的数值解	控制系统	127
abs()	面向矩阵元素的模运算，类似的函数还有 sqrt(), exp(), sin(), cos(), tan(), asin(), acos(), atan(), atan2(), log(), log10(), real(), imag(), conj(), ceil(), floor(), round(), fix() 等	MATLAB	128
expm()	矩阵的指数运算，支持符号运算，其他函数为 expm1(), expm2(), expm3() 等，但不支持符号运算	MATLAB	129
funm()	矩阵函数计算，可以求取任意非线性矩阵函数，不能由于符号运算	MATLAB	132
funm()	矩阵函数计算，可以用符号方法求取任意非线性矩阵函数	自编	135

- 介绍了零矩阵、幺矩阵、单位矩阵、随机数矩阵、对角矩阵、Hilbert 矩阵、伴随矩阵、Vendermonde 矩阵及 Hankel 矩阵等特殊矩阵的 MATLAB 函数，并介绍用 MATLAB 语言的符号运算工具箱语句编写输出符号矩阵的方法。
- 可以利用 MATLAB 语句对给定矩阵进行数值解与解析解分析，如计算矩阵的行列式、迹、秩、范数、特征多项式、逆矩阵和广义逆矩阵、特征值与特征向量等。

- 还介绍了矩阵的分解方法，如 LU 分解、正交分解、对称矩阵的 Cholesky 分解、Jordan 分解、奇异值分解等，介绍利用 MATLAB 语言直接对矩阵分解的数值解和解析解方法。
- 分析了线性代数方程可解的条件，分别对惟一解、无穷解和无解等问题进行处理，给出了基于 MATLAB 语言的无穷解的基础解系与通解求取方法，还介绍了无解方程的最小二乘求解方法等。
- 分析了连续、离散 Lyapunov 方程及 Sylvester 方程的数值解法和基于 Kronecker 乘积的解析解算法，并给出了解析解函数实现，还研究了基于 MATLAB 语言的二次型 Riccati 方程的数值解法。
- 引入了逐点函数求值和矩阵函数求值的概念，并对指数函数、三角函数等矩阵函数求值给出了基于 MATLAB 语言的解析解、数值解方法，并给出了基于 Jordan 分解的矩阵一般矩阵函数求值方法和 MATLAB 程序，可以推出诸如 $e^{\boldsymbol{A}\cos \boldsymbol{A}t}$ 型的任意矩阵函数求值 MATLAB 解析程序，理论上可以求解任意复杂的矩阵函数求值问题。

4.7 习 题

1 Jordan 矩阵是矩阵分析中一类很实用的矩阵，其一般形式为

$$\boldsymbol{J}=\begin{bmatrix}-\alpha & 1 & 0 & \cdots & 0\\ 0 & -\alpha & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & -\alpha\end{bmatrix},\quad 例如\ \boldsymbol{J}_1=\begin{bmatrix}-5 & 1 & 0 & 0 & 0\\ 0 & -5 & 1 & 0 & 0\\ 0 & 0 & -5 & 1 & 0\\ 0 & 0 & 0 & -5 & 1\\ 0 & 0 & 0 & 0 & -5\end{bmatrix}$$

试利用 **diag()** 函数给出构造 $\boldsymbol{J}_1$ 的语句。

2 幂零矩阵是一类特殊的矩阵，其基本形式为

$$\boldsymbol{H}_n=\begin{bmatrix}0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ 0 & 0 & 0 & \cdots & 0\end{bmatrix}$$

亦即，矩阵的次主对角线元素为 1，其余均为 0，试验证对指定阶次的幂零矩阵，有 $\boldsymbol{H}_n^i=\boldsymbol{0}$ 对所有的 $i\geqslant n$ 成立。

3 试从矩阵的显示格式区分符号矩阵和数值矩阵，明确它们的含义和应用场合。若 $\boldsymbol{A}$ 矩阵为数值矩阵，$\boldsymbol{B}$ 为符号矩阵，$\boldsymbol{C}$=$\boldsymbol{A}$*$\boldsymbol{B}$ 运算得出的 $\boldsymbol{C}$ 矩阵是符号矩阵还是数值矩阵？

4 请将下面给出的矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 输入到 MATLAB 环境中，并将它们转换成符号矩阵。

$$\boldsymbol{A}=\begin{bmatrix}5&7&6&5&1&6&5\\2&3&1&0&0&1&4\\6&4&2&0&6&4&4\\3&9&6&3&6&6&2\\10&7&6&0&0&7&7\\7&2&4&4&0&7&7\\4&8&6&7&2&1&7\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}3&5&5&0&1&2&3\\3&2&5&4&6&2&5\\1&2&1&1&3&4&6\\3&5&1&5&2&1&2\\4&1&0&1&2&0&1\\-3&-4&-7&3&7&8&12\\1&-10&7&-6&8&1&5\end{bmatrix}$$

5 试求出 Vandermonde 矩阵 $\boldsymbol{A}=\begin{bmatrix}a^4&a^3&a^2&a&1\\b^4&b^3&b^2&b&1\\c^4&c^3&c^2&c&1\\d^4&d^3&d^2&d&1\\e^4&e^3&e^2&e&1\end{bmatrix}$ 的行列式，并以最简的形式显示结果。

6 利用 MATLAB 语言提供的现成函数对习题 4 中给出的两个矩阵进行分析，判定它们是否为奇异矩阵，得出矩阵的秩、行列式、迹和逆矩阵，检验得出的逆矩阵是否正确。

7 试求出习题 4 中给出的 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵的特征多项式、特征值与特征向量，并验证 Hamilton-Caylay 定理，解释并验证如何运算能消除误差。

8 试对习题 4 中给出的 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵进行奇异值分解、LU 分解及正交分解矩阵。

9 试求出下面矩阵的特征值、特征向量、奇异值。

$$\boldsymbol{A}=\begin{bmatrix}2&7&5&7&7\\7&4&9&3&3\\3&9&8&3&8\\5&9&6&3&6\\2&6&8&5&4\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}703&795&980&137&661\\547&957&271&12&284\\445&523&252&894&469\\695&880&876&199&65\\621&173&737&299&988\end{bmatrix}$$

10 试判定下面矩阵是否为正定矩阵，如果是，则得出其 Cholesky 分解矩阵。

$$\boldsymbol{A}=\begin{bmatrix}9&2&1&2&2\\2&4&3&3&3\\1&3&7&3&4\\2&3&3&5&4\\2&3&4&4&5\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}16&17&9&12&12\\17&12&12&2&18\\9&12&18&7&13\\12&2&7&18&12\\12&18&13&12&10\end{bmatrix}$$

11 试对矩阵 $\boldsymbol{A}=\begin{bmatrix}-2&0.5&-0.5&0.5\\0&-1.5&0.5&-0.5\\2&0.5&-4.5&0.5\\2&1&-2&-2\end{bmatrix}$ 进行 Jordan 变换，并得出变换矩阵。

12 试求下面齐次方程的基础解系。

$$\begin{cases}6x_1+x_2+4x_3-7x_4-3x_5=0\\-2x_1-7x_2-8x_3+6x_4\quad=0\\-4x_1+5x_2+x_3-6x_4+8x_5=0\\-34x_1+36x_2+9x_3-21x_4+49x_5=0\\-26x_1-12x_2-27x_3+27x_4+17x_5=0\end{cases}$$

13 试求下面线性代数方程的解析解与数值解，并检验解的正确性。

$$\begin{bmatrix}2&-9&3&-2&-1\\10&-1&10&5&0\\8&-2&-4&-6&3\\-5&-6&-6&-8&-4\end{bmatrix}\boldsymbol{X}=\begin{bmatrix}-1&-4&0\\-3&-8&-4\\0&3&3\\9&-5&3\end{bmatrix}$$

14 试判定下面的线性代数方程是否有解。

$$\begin{bmatrix}16&2&3&13\\5&11&10&8\\9&7&6&12\\4&14&15&1\end{bmatrix}\boldsymbol{X}=\begin{bmatrix}1\\3\\4\\7\end{bmatrix}$$

15 试求出线性代数方程的解析解，并验证解的正确性。

$$\begin{bmatrix}2&9&4&12&5&8&6\\12&2&8&7&3&3&7\\3&0&3&5&7&5&10\\3&11&6&6&9&9&1\\11&2&1&4&6&8&7\\5&-18&1&-9&11&-1&18\\26&-27&-1&0&-15&-13&18\end{bmatrix}\boldsymbol{X}=\begin{bmatrix}1&9\\5&12\\4&12\\10&9\\0&5\\10&18\\-20&2\end{bmatrix}$$

16 试用数值方法和解析方法求取下面的 Sylvester 方程，并验证得出的结果。

$$\begin{bmatrix}3&-6&-4&0&5\\1&4&2&-2&4\\-6&3&-6&7&3\\-13&10&0&-11&0\\0&4&0&3&4\end{bmatrix}\boldsymbol{X}+\boldsymbol{X}\begin{bmatrix}3&-2&1\\-2&-9&2\\-2&-1&9\end{bmatrix}=\begin{bmatrix}-2&1&-1\\4&1&2\\5&-6&1\\6&-4&-4\\-6&6&-3\end{bmatrix}$$

17 假设某 Riccati 方程的数学表达式为 $\boldsymbol{PA}+\boldsymbol{A}^{\rm T}\boldsymbol{P}-\boldsymbol{PBR}^{-1}\boldsymbol{B}^{\rm T}\boldsymbol{P}+\boldsymbol{Q}=\boldsymbol{0}$，且已知

$$\boldsymbol{A}=\begin{bmatrix}-27&6&-3&9\\2&-6&-2&-6\\-5&0&-5&-2\\10&3&4&-11\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}0&3\\16&4\\-7&4\\9&6\end{bmatrix},\ \boldsymbol{Q}=\begin{bmatrix}6&5&3&4\\5&6&3&4\\3&3&6&2\\4&4&2&6\end{bmatrix},\ \boldsymbol{R}=\begin{bmatrix}4&1\\1&5\end{bmatrix}$$

试求解该方程，得出 $\boldsymbol{P}$ 矩阵，并检验得出解的精度。

18 假设已知某 Jordan 块矩阵 $\boldsymbol{A}$ 及其组成部分为

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{A}_1&&\\&\boldsymbol{A}_2&\\&&\boldsymbol{A}_3\end{bmatrix},\ \boldsymbol{A}_1=\begin{bmatrix}-3&1&0\\0&-3&1\\0&0&-3\end{bmatrix},\ \boldsymbol{A}_2=\begin{bmatrix}-5&1\\0&-5\end{bmatrix},\ \boldsymbol{A}_3=\begin{bmatrix}-1&1&0&0\\0&-1&1&0\\0&0&-1&1\\0&0&0&-1\end{bmatrix}$$

试用解析解运算的方式得出 $\mathrm{e}^{\boldsymbol{A}t}$，$\sin\left(2\boldsymbol{A}t+\frac{\pi}{3}\right)$，$\mathrm{e}^{\boldsymbol{A}^2t}\boldsymbol{A}^2+\sin(\boldsymbol{A}^3t)\boldsymbol{A}t+\mathrm{e}^{\sin\boldsymbol{A}t}$。

19 假设已知矩阵 $\boldsymbol{A}$ 如下，试求出 $\mathrm{e}^{\boldsymbol{A}t},\sin\boldsymbol{A}t,\mathrm{e}^{\boldsymbol{A}t}\sin\left(\boldsymbol{A}^2\mathrm{e}^{\boldsymbol{A}t}t\right)$。

$$\boldsymbol{A}=\begin{bmatrix}-4.5&0&0.5&-1.5\\-0.5&-4&0.5&-0.5\\1.5&1&-2.5&1.5\\0&-1&-1&-3\end{bmatrix}$$

第5章 积分变换与复变函数问题的计算机求解

积分变换技术可以将某些难以分析的问题通过映射的方式映射到其他域内的表达式后再进行分析。例如，Laplace 变换可以将时域函数映射成复域函数，从而可以将某时域函数的微分方程映射成复域的多项式代数方程，使得原微分方程在诸多方面，如稳定性、解析解等方面更便于分析，这样的变换方法构成了经典自动控制理论的基础。在实际应用中，Fourier 变换、Mellin 变换及 Hankel 变换都是有其应用领域的。如何利用计算机求解积分变换的解析解是本章主要介绍的问题之一。第 5.1 节将首先介绍 Laplace 变换与反变换的定义及基本性质，然后介绍用 MATLAB 语言中的符号运算工具箱函数求取 Laplace 变换及反变换问题解析解方法。第 5.2 节将介绍 Fourier 变换及反变换的定义、性质和变换问题的 MATLAB 解法，并介绍 Fourier 余弦变换、正弦变换等问题的计算机求解方法。第 5.3 节将介绍 Mellin 变换、Hankel 变换等问题的 MATLAB 语言的求解算法，可以得出函数的相应变换及反变换。Z 变换是另一类实用的变换方法，该变换方法也是离散控制理论的数学基础。第 5.4 节将介绍 Z 变换及其反变换的定义和性质，并介绍基于 MATLAB 语言符号运算工具箱的 Z 变换问题的计算机辅助求解方法。本章的另一个主要问题是复变函数问题及其 MATLAB 语言求解，可以用第 5.5 节中介绍的方法计算复变函数的奇点，进行部分分式展开等运算，并介绍利用相关方式求解封闭区域积分问题的方法。

5.1 Laplace 变换及其反变换

法国数学家 Pierre-Simon Laplace (1749–1827) 引入的积分变换可以巧妙地将一般常系数微分方程映射成代数方程，奠定了很多领域，如电路分析、自动控制原理等的数学模型基础。本节将首先介绍 Laplace 变换及其反变换的定义与性质，然后介绍利用计算机数学语言 MATLAB 求解 Laplace 变换及其反变换的方法与应用。

5.1.1 Laplace 变换及反变换定义与性质

一个时域函数 $f(t)$ 的 Laplace 变换可以定义为

$$\mathscr{L}[f(t)]=\int_0^\infty f(t)\mathrm{e}^{-st}\mathrm{d}t=F(s) \tag{5-1-1}$$

式中，$\mathscr{L}[f(t)]$ 为 Laplace 变换的简单记号。

下面将不加证明地列出一些 Laplace 变换的性质。

① **线性性质** 若 a 与 b 均为标量，则 $\mathscr{L}[af(t)\pm bg(t)]=a\mathscr{L}[f(t)]\pm b\mathscr{L}[g(t)]$。

② **时域平移性质** $\mathscr{L}[f(t-a)]=\mathrm{e}^{-as}F(s)$。

③ **s-域平移性质** $\mathscr{L}[\mathrm{e}^{-at}f(t)]=F(s+a)$。

④ **微分性质** $\mathscr{L}[\mathrm{d}f(t)/\mathrm{d}t]=sF(s)-f(0^+)$，一般地，$n$ 阶微分可以由下式求出。

$$\mathscr{L}\left[\frac{\mathrm{d}^n}{\mathrm{d}t^n}f(t)\right]=s^nF(s)-s^{n-1}f(0^+)-s^{n-2}\frac{\mathrm{d}f(0^+)}{\mathrm{d}t}-\cdots-\frac{\mathrm{d}^{n-1}f(0^+)}{\mathrm{d}t^{n-1}} \tag{5-1-2}$$

若假设函数 $f(t)$ 及其各阶导数的初值均为 0，则式 (5-1-2) 可以简化成

$$\mathscr{L}[\mathrm{d}^nf(t)/\mathrm{d}t^n]=s^nF(s) \tag{5-1-3}$$

此性质事实上是微分方程映射成代数方程的关键式子。

⑤ **积分性质** 若假设零初始条件，$\mathscr{L}[\int_0^t f(\tau)\mathrm{d}\tau]=F(s)/s$，一般地，函数 $f(t)$ 的 n 重积分的 Laplace 变换可以由下式求出。

$$\mathscr{L}\left[\int_0^t\cdots\int_0^t f(\tau)(\mathrm{d}\tau)^n\right]=\frac{F(s)}{s^n} \tag{5-1-4}$$

⑥ **初值性质** $\lim\limits_{t\to0}f(t)=\lim\limits_{s\to\infty}sF(s)$。

⑦ **终值性质** 如果 $F(s)$ 没有 $s\geqslant0$ 的极点，则 $\lim\limits_{t\to\infty}f(t)=\lim\limits_{s\to0}sF(s)$。

⑧ **卷积性质** $\mathscr{L}[f(t)*g(t)]=\mathscr{L}[f(t)]\mathscr{L}[g(t)]$，式中，卷积算子 $*$ 的定义为

$$f(t)*g(t)=\int_0^t f(\tau)g(t-\tau)\mathrm{d}\tau=\int_0^t f(t-\tau)g(\tau)\mathrm{d}\tau \tag{5-1-5}$$

⑨ **其他性质**

$$\mathscr{L}[t^nf(t)]=(-1)^n\frac{\mathrm{d}^nF(s)}{\mathrm{d}s^n},\ \mathscr{L}\left[\frac{f(t)}{t^n}\right]=\int_s^\infty\cdots\int_s^\infty F(s)\mathrm{d}s^n \tag{5-1-6}$$

如果已知函数的 Laplace 变换式 $F(s)$，则可以通过下面的反变换公式求出其 Laplace 反变换

$$f(t)=\mathscr{L}^{-1}[F(s)]=\frac{1}{2\mathrm{j}\pi}\int_{\sigma-\mathrm{j}\infty}^{\sigma+\mathrm{j}\infty}F(s)\mathrm{e}^{st}\mathrm{d}s \tag{5-1-7}$$

其中 σ 大于 $F(s)$ 奇点的实部，奇点的概念将在后面给出。

5.1.2 Laplace 变换的计算机求解

求解已知函数的 Laplace 变换用数值编程的方式是无法实现的，如果需要采用计算机来求解这样的问题，必须借助于计算机数学语言，如本书介绍的 MATLAB 语言。MATLAB 语言的符号运算工具箱可以轻松地求解 Laplace 变换问题。具体的变换及反变换问题的求解步骤为：

① 定义符号变量 t，这样就能描述时域表达式 Fun 了。和前面介绍的内容一样，申明符号变量可以用 **syms** 命令实现。

② 直接调用 laplace() 函数，就可以求解出所需的时域函数 Laplace 变换式子。该函数的调用格式为

F=laplace(Fun)	采用默认的 t 为时域变量
F=laplace(Fun,v,u)	用户指定时域变量 v 和复域变量名 u

还可以考虑采用 simple() 等函数对其进行化简。

③ 对复杂的问题来说，得出的结果形式通常难以阅读，所以需要调用 pretty() 函数或 latex() 函数对结果进行进一步处理。可以在屏幕上或利用 LaTeX 的强大功能将结果用可读性更强的形式显示出来。

如果已知 Laplace 变换式子，则应该首先给出 Laplace 变换式子 Fun，然后采用符号运算工具箱中的 ilaplace() 函数对其进行反变换。该函数的调用格式为

f=laplace(Fun)	采用默认的 s 为时域变量
f=ilaplace(Fun,u,v)	用户指定时域变量 v 和复域变量名 u

获得变化式子之后也可以对之进一步化简和改变显示格式。

【例 5-1】已知函数 $f(t)=t^2\mathrm{e}^{-2t}\sin(t+\pi)$，试求取该函数的 Laplace 变换。

【求解】分析原题，可以先申明 t 为符号变量，再用 MATLAB 语句表示给定的 $f(t)$ 函数，然后就可以用下面的语句立即得出该函数的 Laplace 变换为

```
>> syms t; f=t^2*exp(-2*t)*sin(t+pi); laplace(f)
ans =
   -8/((s+2)^2+1)^3*(s+2)^2+2/((s+2)^2+1)^2
```

上述结果的可读性不是很好，所以可以考虑采用 pretty() 函数修改显示形式为

```
>> pretty(ans)

                         2
                  (s + 2)              2
          -8 ---------------- + ---------------
                     2     3            2     2
              ((s + 2)  + 1)    ((s + 2)  + 1)
```

如果用户采用 LaTeX 排版语言进行文字处理，则可以调用 latex(ans) 函数进行处理，并将结果嵌入 LaTeX 排版语言。可以得出下面的排版效果：

$$-8\frac{(s+2)^2}{\left((s+2)^2+1\right)^3}+2\frac{1}{\left((s+2)^2+1\right)^2}$$

【例 5-2】假设给出的原函数为 $f(x)=x^2\mathrm{e}^{-2x}\sin(x+\pi)$，试求其 Laplace 变换，并对结果进行 Laplace 反变换，看是否能变换回原函数。

【求解】同样可以采用 laplace()。

```
>> syms x w; f=x^2*exp(-2*x)*sin(x+pi);
   F=laplace(f,x,w)
F =
```

```
    -8/((w+2)^2+1)^3*(w+2)^2+2/((w+2)^2+1)^2
```

可见，这样的结果和上面的例子是完全一致的，但需要按要求做一下变量替换。

使用 Laplace 反变换的函数 ilaplace() 得出原函数，因为 $\sin(t+\pi)=-\sin t$。

```
>> ilaplace(F)
ans =
    -t^2*exp(-2*t)*sin(t)
```

【例 5-3】 试求出下面函数的 Laplace 反变换。

$$G(x)=\frac{-17x^5-7x^4+2x^3+x^2-x+1}{x^6+11x^5+48x^4+106x^3+125x^2+75x+17}$$

【求解】 从原来给出的问题，似乎用下面的语句就能直接求解出所需结果。

```
>> syms x t;
   G=(-17*x^5-7*x^4+2*x^3+x^2-x+1)/(x^6+11*x^5+48*x^4+106*x^3+125*x^2+75*x+17);
   f=ilaplace(G,x,t)
f =
    -1/31709*sum((39275165+45806941*_alpha^4+156459285*_alpha+5086418*_alpha^5+
    149142273*_alpha^3+221566216*_alpha^2)*exp(_alpha*t),
    _alpha = RootOf(_Z^6+11*_Z^5+48*_Z^4+106*_Z^3+125*_Z^2+75*_Z+17))
```

而事实上，该方法并不能求出有意义的结果。这是因为分母多项式对应的方程的解不存在解析解，所以导致原问题不存在解析解。若利用 MATLAB 的变精度算法，则可以得出高精度的数值解。

```
>> vpa(f,16)
```

得出的结果可以用 LaTeX 表示成

$$\begin{aligned}
g(t)=&-556.2565306869612\mathrm{e}^{-3.261731010738519t}+0.2125567969626323\mathrm{e}^{-0.5208596052932005t}+\\
&537.2850440376093\mathrm{e}^{-2.530945820048848t}\cos(0.3997631054498554t)-\\
&698.2462189990537\mathrm{e}^{-2.530945820048848t}\sin(0.3997631054498554t)-\\
&0.000031536787662808 67\mathrm{j}\left(11070344.67912050\mathrm{e}^{-2.530945820048848t}\cos(0.3997631054498554t)\right.\\
&\quad\left.+8518385.730694274\mathrm{e}^{-2.530945820048848t}\sin(0.3997631054498554t)\right)-\\
&0.00003153678766280867\mathrm{j}\left(-11070344.67912050\mathrm{e}^{-2.530945820048848t}\cos(0.3997631054498554t)\right.\\
&\quad\left.-8518385.730694274\mathrm{e}^{-2.530945820048848t}\sin(0.3997631054498554t)\right)+\\
&1.758929852389242\mathrm{e}^{-1.077758871935292t}\cos(0.6021065910607615t)+\\
&10.99415251571466\mathrm{e}^{-1.077758871935292t}\sin(0.6021065910607615t)-\\
&0.00003153678766280867\mathrm{j}\left(-174306.7910603980\mathrm{e}^{-1.077758871935292t}\cos(0.6021065910607615t)\right.\\
&\quad\left.+27886.95334470523\mathrm{e}^{-1.077758871935292t}\sin(0.6021065910607615t)\right)-\\
&0.00003153678766280867\mathrm{j}\left(174306.7910603980\mathrm{e}^{-1.077758871935292t}\cos(0.6021065910607615t)\right.\\
&\quad\left.-27886.95334470523\mathrm{e}^{-1.077758871935292t}\sin(0.6021065910607615t)\right)
\end{aligned}$$

注意，由于这里涉及高阶多项式方程的求解，而该方程本身并没有解析解，所以这个问题也只能求出高精度的数值解，但无法求出解析解，因为解析解根本不存在。

【例 5-4】对例 5-1 给出的原函数 $f(t)$，试得出 $\mathscr{L}[\mathrm{d}^5 f(t)/\mathrm{d}t^5]$ 和 $s^5\mathscr{L}[f(t)]$ 之间的关系。

【求解】如果想求解这样的问题，可以利用符号运算工具箱中的 diff() 函数对函数 $f(t)$ 求五阶导数，再进行 Laplace 变换，则

```
>> syms t s; f=t^2*exp(-2*t)*sin(t+pi);
   F=simple(laplace(diff(f,t,5)))
ans =
   -2*(3000+6825*s+110*s^5+6660*s^2+960*s^4+3471*s^3)/(s^2+4*s+5)^3
```

对 $f(t)$ 进行 Laplace 变换，并将变换结果乘以 s^5，将得出的结果与前面直接得出的结果相减，可以得到

```
>> F0=laplace(f); simple(F-s^5*F0)
ans =
   6*s-48
```

由于二者之差不为 0，所以看起来和式 (5-1-3) 是不同的。这是因为 $f(t)$ 函数有 $f(0) = f'(0) = 0$，但其高阶导数在 $t = 0$ 处的值不为 0，故不满足式 (5-1-3)，而满足式 (5-1-2)。由式 (5-1-2) 直接可见，考虑初始条件后，得出的差和上述的差完全一致。

```
>> ss=0; f1=f;
   for i=4:-1:0
      ss=ss-s^i*subs(f1,t,0); f1=diff(f1,t);
   end
ss =
   6*s-48
```

【例 5-5】试推导出 $\mathscr{L}[\mathrm{d}^2 f(t)/\mathrm{d}t^2]$ 的微分公式。

【求解】MATLAB 的符号运算工具箱还可以进行一些简单的 Laplace 变换公式推导。假设想导出 $f(t)$ 的二阶导数的 Laplace 变换，首先应该先定义一下 $f(t)$ 函数，这可以通过如下语句实现，并推导出二阶导数的 Laplace 变换公式。

```
>> syms t; y=sym('f(t)')  % 定义原函数
   laplace(diff(y,t,2))
ans =
   s*(s*laplace(f(t),t,s)-f(0))-D(f)(0)
```

当然，该功能可以进一步引申，求出函数 8 阶导数的 Laplace 变换。

```
>> laplace(diff(y,t,8))
ans =
   s*(s*(s*(s*(s*(s*(s*(s*laplace(f(t),t,s)-f(0))-D(f)(0))-@@(D,2)(f)(0))-
   @@(D,3)(f)(0))-@@(D,4)(f)(0))-@@(D,5)(f)(0))-@@(D,6)(f)(0))-@@(D,7)(f)(0)
```

【例 5-6】已知 $f(t) = \mathrm{e}^{-5t}\cos(2t+1)+5$，试求出 $\mathscr{L}[\mathrm{d}^5 f(t)/\mathrm{d}t^5]$。

【求解】这个例子是上个例子的引申。若已知某具体函数 $f(t)$，则可以将 diff() 函数与 laplace() 函数结合起来使用，这样用下面的 MATLAB 命令则可以得出所需的结果。

```
>> syms t; f=exp(-5*t)*cos(2*t+1)+5;
```

```
F=laplace(diff(f,t,5)); F=simple(F); latex(F)
```

则其结果可以在 LaTeX 环境中显示为

$$\frac{1475\cos 1s - 1189\cos 1 - 24360\sin 1 - 4282\sin 1s}{s^2+10s+29}$$

对化简后的结果其实还可以采用其他化简方法微调。例如，想将分子多项式合并同类项，则可以给出如下语句：

```
>> syms s; collect(F); latex(ans)
```

则将得出如下的显示结果：

$$\frac{(1475\cos 1 - 4282\sin 1)\,s - 1189\cos 1 - 24360\sin 1}{s^2+10s+29}$$

5.2 Fourier 变换及其反变换

5.2.1 Fourier 变换及反变换定义与性质

Fourier 变换的一般定义为

$$\mathscr{F}[f(t)] = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t = F(\omega) \tag{5-2-1}$$

如果已知 Fourier 变换式子 $F(\omega)$，则可以由 Fourier 反变换公式反演出 $f(t)$ 函数为

$$f(t) = \mathscr{F}^{-1}[F(\omega)] = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} F(\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega \tag{5-2-2}$$

这里仍将不加证明地列出一些 Fourier 变换的性质。

① **线性性质** 若 a 与 b 均为标量，则 $\mathscr{F}[af(t)\pm bg(t)] = a\mathscr{F}[f(t)] \pm b\mathscr{F}[g(t)]$。

② **平移性质** $\mathscr{F}[f(t\pm a)] = \mathrm{e}^{\pm\mathrm{j}a\omega}F(\omega)$。

③ **复域平移性质** $\mathscr{F}[\mathrm{e}^{\pm\mathrm{j}at}f(t)] = F(\omega\mp a)$。

④ **微分性质** $\mathscr{F}[\mathrm{d}f(t)/\mathrm{d}t] = \mathrm{j}\omega F(\omega)$，一般地，$n$ 阶微分可以由下式求出。

$$\mathscr{F}\left[\frac{\mathrm{d}^n}{\mathrm{d}t^n}f(t)\right] = (\mathrm{j}\omega)^n\mathscr{F}[f(t)] \tag{5-2-3}$$

⑤ **积分性质** $\mathscr{F}\left[\int_{-\infty}^{t} f(\tau)\mathrm{d}\tau\right] = F(\omega)/(\mathrm{j}\omega)$，一般地，函数 $f(t)$ 的 n 重积分的 Fourier 变换可以由下式求出。

$$\mathscr{F}\left[\int_{-\infty}^{t}\cdots\int_{-\infty}^{t} f(\tau)(\mathrm{d}\tau)^n\right] = \frac{\mathscr{F}[f(t)]}{(\mathrm{j}\omega)^n} \tag{5-2-4}$$

⑥ **尺度变换** $\mathscr{F}[f(at)] = \dfrac{1}{a}F\left(\dfrac{\omega}{a}\right)$。

⑦ **卷积性质** $\mathscr{F}[f(t)*g(t)] = \mathscr{F}[f(t)]\mathscr{F}[g(t)]$，其中，卷积定义仍为式 (5-1-5)。

5.2.2 Fourier 变换的计算机求解

和 Laplace 变换一样，应该先申明符号变量，并定义出原函数为 Fun，这样就可以按如下的格式调用 Fourier 变换求解函数 fourier()，得出该函数的 Fourier 变换式。

F=fourier(Fun)	按默认变量进行 Fourier 变换
F=fourier(Fun, v, u)	将 v 的函数变换成 u 的函数

注意，和式 (5-2-1) 不同，MATLAB 符号运算工具箱中 fourier() 函数定义的连续信号 $f(x)$ 的 Fourier 变换定义为

$$\mathscr{F}[f(x)]=\int_{-\infty}^{\infty}f(x)\mathrm{e}^{-\mathrm{j}\omega x}\mathrm{d}x=F_1(\omega) \tag{5-2-5}$$

可见，二者相差是 $\sqrt{2\pi}$ 的关系。得出了 MATLAB 定义的 Fourier 变换 $F_1(\omega)$，就能用 $F_1(\omega)/\sqrt{2\pi}$ 得出信号的 Fourier 变换表达式。

给出了 Fourier 变换表达式，则可以通过 ifourier() 函数求解该函数的 Fourier 反变换问题。该函数的具体调用格式为

f=ifourier(Fun)	按默认变量进行 Fourier 反变换
f=ifourier(Fun, u, v)	将 u 的函数变换成 v 的函数

同样，该工具箱中定义的 Fourier 变换的反变换公式也和式 (5-2-2) 有区别。MATLAB 中给定函数 $F(\omega)$ 的 Fourier 反变换公式为

$$f(x)=\mathscr{F}^{-1}[F_1(\omega)]=\frac{1}{2\pi}\int_{-\infty}^{\infty}F_1(\omega)\mathrm{e}^{\mathrm{j}\omega x}\mathrm{d}\omega \tag{5-2-6}$$

可见，该公式和式 (5-2-6) 中给出定义完全一致。所以在使用 MATLAB 符号运算工具箱求解时应该注意二者的差异，以免出现不必要的错误。

从上面的语句可以看出，如果定义了已知函数，则可以用一个语句求解出其 Fourier 变换或反变换式子，变换函数的调用和 Laplace 变换一样简单。所以如果已知数学定义下的 Fourier 变换式子 $F(\omega)$，则可以将其先除以 $\sqrt{2\pi}$，再进行变换。如果已知的是 MATLAB 得出的 Fourier 变换式子，则可以直接使用 ifourier() 函数进行变换。

【例 5-7】 考虑 $f(t)=1/(t^2+a^2)$, $a>0$，试写出该函数的 Fourier 变换式。

【求解】 可以用下面的语句得出原函数的 Fourier 变换。

```
>> syms t w; syms a potitive
   f=1/(t^2+a^2); F=fourier(f,t,w)
F =
   pi*(exp(-a*w)*Heaviside(w)+exp(a*w)*Heaviside(-w))/a
```

其中得出的 Heaviside(ω) 函数为 ω 的阶跃函数，又称为 Heaviside 函数，当 $\omega\geqslant 0$ 时，该函数的值为 1，否则为 0，而当 $\omega\leqslant 0$ 时，Heaviside($-\omega$) 的值为 1，否则为 0。假设 $\omega>0$，则 F 可以简化成 $\pi\mathrm{e}^{-a\omega}/a$，若 $\omega<0$，则 F 可以简化成 $\pi\mathrm{e}^{a\omega}/a$，故可以将上述结果写成

$$\mathscr{F}[f(t)]=\frac{\pi\mathrm{e}^{-a|\omega|}}{a}$$

然而，这样的最简表达式是通过 MATLAB 语言的自动化简功能无法得出的。对得出的结果进行 Fourier 反变换，则将还原出原函数。

```
>> syms t w; syms a positive
   f=pi*exp(-a*abs(w))/a; ifourier(f)
ans =
    1/(a^2+x^2)
```

即使不手工化简，仍可以直接对前面得出的 F 函数进行反变换，仍将得出原函数。

```
>> ifourier(F)
ans =
    1/(a^2+x^2)
```

【例 5-8】假设时域函数为 $f(t)=\sin^2(at)/t$，$a>0$，试求出其 Fourier 变换。

【求解】由给定的式子，可以用下面的语句获得原函数的 Fourier 变换。

```
>> syms t w; syms a positive
   f=sin(a*t)^2/t; fourier(f,t,w)
ans =
   1/2*i*pi*(Heaviside(w-2*a)+Heaviside(w+2*a)-2*Heaviside(w))
```

可见，该结果仍然依赖于 Heaviside() 函数，所以理解 Heaviside() 函数将使得用户能得到更简单的形式。当 $\omega>2a$，则 3 个 Heaviside() 函数的值均为 1，故 $F(\omega)=0$。若 $\omega\leqslant-2a$，则 3 个函数的值均为 0，故 $F(\omega)=0$。若 $0<\omega<2a$，则第 2 和 3 个 Heaviside() 的值为 1，故 $F(\omega)=-\mathrm{j}\pi/2$。当 $0>\omega>-2a$，则 $F(\omega)=\mathrm{j}\pi/2$。综上所述，原函数的 Fourier 变换可以化简成

$$\mathscr{F}[f(t)]=\begin{cases}0, & |\omega|>2a\\ -\mathrm{j}\pi\mathrm{sign}(\omega)/2, & |\omega|<2a\end{cases}$$

【例 5-9】再考虑一个稍微复杂的例子。假设函数为 $f(t)=\mathrm{e}^{-a|t|}/\sqrt{|t|}$，试用 MATLAB 提供的现成函数和直接积分的方法分别求解 Fourier 变换问题。

【求解】先考虑用现成的函数 fourier() 对给出的原函数进行变换，可以采用下面的语句求取其 Fourier 变换。

```
>> syms w t; syms a positive
   f=exp(-a*abs(t))/sqrt(abs(t)); F=fourier(f,t,w)
F =
   fourier(exp(-a*abs(t))/abs(t)^(1/2),t,w)
```

很遗憾，该函数无法求取原函数的 Fourier 变换式子。还可以考虑从底层进行积分计算，将式 (5-2-1) 给出的积分式分成两段分别求积分，亦即 $\int_{-\infty}^{\infty}=\int_{-\infty}^{0}+\int_{0}^{\infty}$，这样可以给出如下的 MATLAB 命令，但这样的方法也无法得出该函数的 Fourier 变换。

```
>> f1=exp(a*t)/sqrt(-t); f2=exp(-a*t)/sqrt(t); j=sym(sqrt(-1));
   F=int(f1*exp(-j*w*t),-inf,0)+int(f2*exp(-j*w*t),0,inf)
ans =
   undefined
```

事实上，上述原函数是有有限 Fourier 变换的，但由于 MATLAB 和 Maple 本身存在的困难，目前无法求取该函数的 Fourier 变换。

5.2.3 Fourier 正弦和余弦变换

Fourier 正弦变换的一般定义为

$$\mathscr{F}_{\mathscr{S}}[f(t)]=\int_0^{\infty}f(t)\sin(\omega t)\mathrm{d}t=F_s(\omega) \tag{5-2-7}$$

Fourier 余弦变换的一般定义为

$$\mathscr{F}_{\mathscr{C}}[f(t)]=\int_0^{\infty}f(t)\cos(\omega t)\mathrm{d}t=F_c(\omega) \tag{5-2-8}$$

相应地，还可以如下定义 Fourier 正弦和余弦反变换为

$$\mathscr{F}_{\mathscr{S}}^{-1}[F_s(t)]=\frac{2}{\pi}\int_{-\infty}^{\infty}F_s(\omega)\sin(\omega t)\mathrm{d}\omega \tag{5-2-9}$$

$$\mathscr{F}_{\mathscr{C}}^{-1}[F_c(t)]=\frac{2}{\pi}\int_{-\infty}^{\infty}F_c(\omega)\cos(\omega t)\mathrm{d}\omega \tag{5-2-10}$$

类似于 Fourier 变换，还可以定义出对称的 Fourier 正弦和余弦变换。在正变换中应该乘以 $\sqrt{2/\pi}$，反变换应该除以 $\sqrt{2/\pi}$，故若能得到前面定义的变换，就可以转换成对称定义的变换。

MATLAB 语言的符号运算工具箱中并未直接提供余弦 Fourier 变换的函数，所以可以考虑采用符号积分的方法直接求取余弦 Fourier 变换。下面将提供具体例子演示正弦和余弦 Fourier 变换的推导方法。

【例 5-10】 试求出 $f(t)=t^n\mathrm{e}^{-at},a>0,n=1,2,\cdots,8$ 的余弦 Fourier 变换。

【求解】 解决这样的问题可以采用循环结构，对不同的 i 值，可以用直接积分的算法求取 Fourier 余弦变换，并将得出的结果进行化简，再转换 LaTeX 格式列出，如表 5-1 所示。

```
>> syms t w; syms a positive
   for i=1:8
      f=t^i*exp(-a*t); F=int(f*cos(w*t),t,0,inf); latex(simple(F))
   end
```

其实，按照数学手册[41]中给出的公式，对整数 n，可以得出

$$\mathscr{F}_{\mathscr{C}}\left[t^n\mathrm{e}^{-at}\right]=n!\left(\frac{a}{a^2+\omega^2}\right)^{n+1}\sum_{m=0}^{[n/2]}(-1)^m C_{n+1}^{2m+1}\left(\frac{\omega}{a}\right)^{2m+1} \tag{5-2-11}$$

MATLAB 语言并未直接提供 Fourier 正弦、余弦变换的现成函数，而 Maple 提供了 Fourier 正弦、余弦变换的函数，故可以用 MATLAB 的符号运算工具箱直接调用 Maple 中的现成函数。具体格式为

表 5-1 n 取不同值时 Fourier 余弦变换结果

n 值	$\mathscr{F}_{\mathscr{C}}[f(t)]$
1~4	$\dfrac{a^2-\omega^2}{(a^2+\omega^2)^2}$, $-2\dfrac{(-a^2+3\omega^2)a}{(a^2+\omega^2)^3}$, $6\dfrac{(-a^2+2a\omega+\omega^2)(-a^2-2a\omega+\omega^2)}{(a^2+\omega^2)^4}$, $24\dfrac{a(a^4-10a^2\omega^2+5\omega^4)}{(a^2+\omega^2)^5}$
5,6	$-120\dfrac{(-a+\omega)(a+\omega)(a^2-4\omega a+\omega^2)(a^2+4\omega a+\omega^2)}{(a^2+\omega^2)^6}$, $-720\dfrac{(-a^6+21a^4\omega^2-35\omega^4a^2+7\omega^6)a}{(a^2+\omega^2)^7}$
7	$5040\dfrac{(a^4+4a^3\omega-6a^2\omega^2-4a\omega^3+\omega^4)\left(a^4-4a^3\omega-6a^2\omega^2+4a\omega^3+\omega^4\right)}{(a^2+\omega^2)^8}$
8	$40320\dfrac{a\left(-a^2+3\omega^2\right)\left(-a^6+33a^4\omega^2-27a^2\omega^4+3\omega^6\right)}{(a^2+\omega^2)^9}$

```
F=maple('fouriersin',f,t,ω)      求解 Fourier 正弦变换
F=maple('fouriercos',f,t,ω)      求解 Fourier 余弦变换
f=maple('invfouriersin',F,ω,t)   求解 Fourier 反正弦变换
f=maple('invfouriercos',F,ω,t)   求解 Fourier 反余弦变换
```

其中，maple() 函数是 MATLAB 符号运算工具箱中的函数，允许用户直接调用 Maple 中的有关函数，'fouriersin' 等是 Maple 中的函数名，其余参数是 Maple 中参数的格式。

【例 5-11】重新考虑例 5-10 中给出的分段函数，令 $n=6$，采用 Maple 中提供的现成函数求解 Fourier 余弦变换问题，并试着对结果进行 Fourier 余弦反变换。

【求解】利用 Maple 中的 Fourier 余弦变换函数和反变换函数可以得出如下的结果：

```
>> syms t w; syms a positive
   f=t^6*exp(-a*t); F=maple('fouriercos',f,t,w); latex(F)
```

该结果可以直接表示为

$$\mathscr{F}_{\mathscr{C}}[f(t)]=720\sqrt{\frac{2}{\pi}}\frac{a^7}{(a^2+\omega^2)^7}\left(1-21\frac{\omega^2}{a^2}+35\frac{\omega^4}{a^4}-7\frac{\omega^6}{a^6}\right)$$

和表 5-1 的结果相比可见，由这种方法得出的不是式 (5-2-8) 中给出的变换，而是对称处理的变换结果，亦即将该式得出的结果乘以 $\sqrt{2/\pi}$ 后得出的结果。使用反变换调用函数则可以还原成原函数。

```
>> f1=maple('invfouriercos',F,w,t)
f1=
   invfouriercos(720*2^(1/2)/pi^(1/2)*a^7/(a^2+w^2)^7*
      (1-21*w^2/a^2+35*w^4/a^4-7*w^6/a^6),w,t)
```

遗憾的是，用其中提供的反变换函数仍无法得出相应的反变换式子。

【例 5-12】试求取下面分段函数的 Fourier 余弦变换。

$$f(t)=\begin{cases}\cos(t), & 0<x<a\\ 0, & 其他\end{cases}$$

【求解】 用 Maple 中变换方法的难处在于表达分段函数，所以仍可以考虑直接通过定义进行变换的方法。研究定义式子可以立即发现，式 (5-2-8) 的被积函数在 $t\in(a,\infty)$ 区间的值为 0，这样，其积分亦为 0，故整个积分问题就变成 $t\in(0,a)$ 区间的积分问题了，故可以用下面的语句求出该函数的 Fourier 余弦变换为

```
>> syms t w; syms a positive
   f=cos(t); F=simple(int(f*cos(w*t),t,0,a)); latex(F)
```

下面列出得出的结果为

$$\mathscr{F}_{\mathscr{C}}[f(t)]=\frac{(1+\omega)\sin(-a+a\omega)+(-1+\omega)\sin(a+a\omega)}{-2+2\omega^2}$$

该式子还可以进一步手工化简为

$$\mathscr{F}_{\mathscr{C}}[f(t)]=\frac{\sin(-a+a\omega)}{2(1+\omega)}+\frac{\sin(a+a\omega)}{2(1-\omega)}$$

5.2.4 离散 Fourier 正弦、余弦变换

离散 Fourier 正弦、余弦变换又称为有限 Fourier 正弦、余弦变换，和前面介绍的 Fourier 正弦、余弦变换相比，其积分区间从 $t\in(0,\infty)$ 变成了 $t\in(0,a)$，故其定义为

$$F_s(k)=\int_0^a f(t)\sin\frac{k\pi t}{a}\mathrm{d}t,\ \ F_c(k)=\int_0^a f(t)\cos\frac{k\pi t}{a}\mathrm{d}t \tag{5-2-12}$$

相应地，可以定义出有限 Fourier 正弦、余弦反变换为

$$f(t)=\frac{2}{a}\sum_{k=1}^{\infty}F_s(k)\sin\frac{k\pi t}{a} \tag{5-2-13}$$

$$f(t)=\frac{1}{a}F_c(0)+\frac{2}{a}\sum_{k=1}^{\infty}F_c(k)\cos\frac{k\pi t}{a} \tag{5-2-14}$$

和前面定义的不同，反变换不再是积分式子，而是无穷级数的求和。下面通过例子介绍如何用 MATLAB 及其符号运算工具箱求取给定函数的有限变换。

【例 5-13】 考虑下面的分段函数

$$f(t)=\begin{cases}t & t\leqslant a/2\\ a-t & t>a/2\end{cases}$$

其中，$a>0$，试求其离散 Fourier 正弦变换。

【求解】 函数的离散 Fourier 正弦变换可以由下面的语句直接求出。

```
>> syms t k; syms a positive
   f1=t; f2=a-t;
   Fs=int(f1*sin(k*pi*t/a),t,0,a/2)+int(f2*sin(k*pi*t/a),t,a/2,a);
   simple(Fs)
```

因为符号运算工具箱不直接支持整数变量，所以该方程只能最终化简成

$$\mathscr{F}_{\mathscr{S}}[f(t)] = \frac{a^2(2\sin(1/2k\pi) - \sin(k\pi))}{k^2\pi^2}$$

考虑到 k 为整数，故有 $\sin(k\pi) \equiv 0$，该式可以手工化简为

$$\mathscr{F}_{\mathscr{S}}[f(t)] = \frac{2a^2}{k^2\pi^2}\sin\frac{k\pi}{2}$$

5.3 其他积分变换问题及求解

除了 Laplace 变换和各种 Fourier 变换外，在不同的领域还需要各种各样其他的变换，如 Mellin 变换、Hankel 变换等。标准的 MATLAB 符号运算工具箱中未直接提供求解这些变换的现成函数，所以解决这些问题仍然有两种方法，其一是采用直接积分的方法，另一种是采用 Maple 语言中的相应函数直接求解。

5.3.1 Mellin 变换

Mellin 变换可以定义为

$$\mathscr{M}[f(x)] = \int_0^\infty f(x)x^{z-1}\mathrm{d}x = M(z) \tag{5-3-1}$$

相应地，Mellin 反变换可以定义为

$$f(x) = \mathscr{M}^{-1}[M(z)] = \frac{1}{2\pi\mathrm{j}}\int_{c-\mathrm{j}\infty}^{c+\mathrm{j}\infty} M(z)x^{-z}\mathrm{d}z \tag{5-3-2}$$

MATLAB 符号运算工具箱并没有直接可用的 Mellin 正反变换的函数，但仍可以通过积分的形式计算 Mellin 变换。下面将通过例子解决这类问题。

【例 5-14】考虑时域函数 $f(t) = \ln t/(t+a)$，其中 $a > 0$，试求其 Mellin 变换。

【求解】根据定义可以用下面语句求取该函数的 Mellin 变换。

```
>> syms t z; syms a positive;
   f=log(t)/(t+a); M=simple(int(f*t^(z-1),t,0,inf)), latex(M)
```

经过化简，可以立即得出如下结果。

$$\mathscr{M}[f(t)] = a^{z-1}\pi\left[\ln a\sin(\pi z) - \pi\cos(\pi z)\right]\csc^2(\pi z)$$

【例 5-15】假设给定函数 $f(t) = 1/(t+a)^n, (a > 0)$，对若干个 n 值求取 Mellin 变换，并总结出对一般 n 值的规律。

【求解】下面的语句将给出 $n = 1, 2, \cdots, 8$ 时 Mellin 变换的结果。

```
>> syms t z; syms a positive
   for i=1:8
      f=1/(t+a)^i; disp(int(f*t^(z-1),t,0,inf))
```

```
    end
  a^(z-1)*pi*csc(pi*z)
  -a^(-2+z)*pi*(z-1)*csc(pi*z)
  1/2*a^(-3+z)*pi*(-2+z)*(z-1)*csc(pi*z)
  -1/6*a^(z-4)*pi*(z-1)*(-2+z)*(-3+z)*csc(pi*z)
  1/24*a^(-5+z)*pi*(z-4)*(-3+z)*(-2+z)*(z-1)*csc(pi*z)
  -1/120*a^(-6+z)*pi*(-5+z)*(z-4)*(-3+z)*(-2+z)*(z-1)*csc(pi*z)
  1/720*a^(-7+z)*pi*(-6+z)*(-5+z)*(z-4)*(-3+z)*(-2+z)*(z-1)*csc(pi*z)
  -1/5040*a^(-8+z)*pi*(-7+z)*(-6+z)*(-5+z)*(z-4)*(-3+z)*(-2+z)*(z-1)*csc(pi*z)
```

所以，可以总结出一般的 Mellin 变换规律为

$$\mathscr{M}\left[\frac{1}{(t+a)^n}\right]=\frac{(-1)^{k-1}\pi}{(n-1)!}a^{z-n}\prod_{i=1}^{n-1}(z-i)\csc(\pi z)$$

Mellin 正反变换问题还可以考虑利用 Maple 语言中的函数 `mellin()` 和 `invmellin()` 求解，这些函数在符号运算工具箱中未给出 MATLAB 实现，但可以通过其支持的语句直接调用 Maple 语言中现成的函数得出所需的结果。这两个函数的具体调用格式为

`F=maple('mellin',f,t,z)`	求解 Mellin 变换
`f=maple('invmellin',F,z,t)`	求解 Mellin 反变换

【例 5-16】仍考虑例 5-15 中给出的函数，若 $n=8$，试利用 Maple 中的相应函数求取该函数的 Mellin 变换，并对结果进行反变换。

【求解】利用 Maple 语言中提供的函数，可以用下面的语句直接得出该函数的 Mellin 变换。

```
>> syms t z; syms a positive
   F=maple('mellin',1/(t+a)^8,t,z)
F =
   -1/5040*pi*(z-1)!/(-8+z)!*a^(-8+z)*csc(pi*z)
>> f1=maple('invmellin',F,z,t)
```

这里的反变换命令得出了一个很长的结果，但仍无法变换回原来给定的函数，也无法得出原问题的有意义的解。

5.3.2 Hankel 变换及求解

Hankel 变换是另一类常用的数学变换，ν 阶 Hankel 变换的数学表达式为

$$\mathscr{H}[f(t)]=\int_0^\infty tf(t)J_\nu(\omega t)\mathrm{d}t=H_\nu(\omega) \tag{5-3-3}$$

其中，$J_\nu(\cdot)$ 为 Bessel 函数。还可以定义 ν 阶 Hankel 反变换公式为

$$\mathscr{H}^{-1}[H(\omega)]=\int_0^\infty \omega H_\nu(\omega)J_\nu(\omega t)\mathrm{d}\omega \tag{5-3-4}$$

可以借助于 Maple 中的 hankel() 和 invhankel() 函数求取给定函数的 Hankel 正反变换，其调用格式为

```
F=maple('hankel',f,t,ω,ν)       求解 Hankel 变换
f=maple('invhankel',F,ω,t,ν)    求解 Hankel 反变换
```

【例 5-17】求取 $f(t)=t^a$ 函数的 0 阶 Hankel 变换。

【求解】利用 Maple 语言中的现成函数，可用用下面语句在 MATLAB 环境中立即写出 Hankel 变换的结果。

```
>> syms t w; syms a positive
   f=t^a; F=maple('hankel',f,t,w,0)
```

即可以得出如下的变换结果：

$$\mathscr{H}[f(t)]=\frac{\omega^{-1-a}\sin\left(1/4\pi\left(2a+3\right)\right)2^{a+1/2}\left(\Gamma\left(1/2a+3/4\right)\right)^2}{\pi}$$

利用 Maple 中现有的 Hankel 变换功能，目前只适合于求解 t^a 型函数的正 Hankel 变换，其他函数的 Hankel 变换用现有的函数不一定能计算出最简的解析结果。例如 $f(t)=a/(a^2+t^2)^{3/2}$ 这样的函数用现有的函数求解 0 阶 Hankel 变换，则有

```
>> syms t x a;
   F=maple('hankel',a/(a^2+t^2)^(3/2),t,w,0)
```

将得出如下的结果：

$$\frac{-9\sqrt{\omega}\mathrm{Hyp}\left([3/4],[1/4,1],1/4\,a^2\omega^2\right)\left(\Gamma\left(3/4\right)\right)^4-4\omega^2\pi^{3/2}\mathrm{Hyp}\left([3/2],[7/4,7/4],1/4\,a^2\omega^2\right)a^{3/2}}{\sqrt{a}\sqrt{\pi}\left(9\Gamma\left(3/4\right)\right)^2}$$

其中，$\mathrm{Hyp}(\cdot)$ 为广义超几何函数，其内容超出本书范围，在此不予详细介绍。而事实上，$f(t)$ 函数的 0 阶 Hankel 变换即为 $\mathrm{e}^{-a\omega}$，故在得出的结果上有很大的差距。所以无论是利用 MATLAB 语言，还是利用现有的 Maple 等计算机数学语言，目前尚不具备进行 Hankel 正反变换的能力，应该用查表的方法得出所需的结果。

5.4 Z 变换及其反变换

5.4.1 Z 变换及反变换定义与性质

离散序列信号 $f(k)$ 的 Z 变换可以定义为

$$\mathscr{Z}[f(k)]=\sum_{i=0}^{\infty}f(k)z^{-k}=F(z) \tag{5-4-1}$$

类似于前面介绍的 Laplace 变换和 Fourier 变换，Z 变换也有很多很多性质，仍不加证明地列出一些性质如下：

① **线性性质** 若 a 与 b 均为标量，则 $\mathscr{Z}[af(k)\pm bg(k)]=a\mathscr{Z}[f(k)]\pm b\mathscr{Z}[g(k)]$。

② **时域平移性质** $\mathscr{Z}[f(k-n)]=z^{-n}F(z)$。

③ **s-域比例性质** $\mathscr{Z}[r^{-k}f(k)]=F(rz)$。

④ **频域微分性质** $\mathscr{Z}[kf(k)]=-z\mathrm{d}F(z)/\mathrm{d}z$。

⑤ **频域积分性质** $\mathscr{Z}[f(k)/k]=\int_z^\infty F(\omega)/\omega\mathrm{d}\omega$。

⑥ **初值性质** $\lim\limits_{k\to0}f(k)=\lim\limits_{z\to\infty}F(z)$。

⑦ **终值性质** 如果 $F(z)$ 无单位圆外的极点，则 $\lim\limits_{k\to\infty}f(k)=\lim\limits_{k\to1}(z-1)F(z)$。

⑧ **卷积性质** $\mathscr{Z}[f(k)*g(k)]=\mathscr{Z}[f(k)]\mathscr{Z}[g(k)]$，式中离散信号的卷积算子 $*$ 定义为

$$f(k)*g(k)=\sum_{l=0}^{\infty}f(k)g(k-l) \tag{5-4-2}$$

给定 Z 变换式子 $F(z)$，则其 Z 反变换的数学表示为

$$f(k)=\mathscr{Z}^{-1}[f(k)]=\frac{1}{2\pi\mathrm{j}}\oint F(z)z^{k-1}\mathrm{d}z \tag{5-4-3}$$

5.4.2 Z 变换的计算机求解

利用 MATLAB 的符号运算工具箱，则 Laplace 变换、Z 变换及其反变换可以很容易地求取出来，掌握这样的工具可以免除复杂问题的手工推导，既节省时间也能避免底层的低级错误。利用符号运算工具箱中提供的 `ztrans()` 和 `iztrans()` 函数可用得出给定函数的正反 Z 变换。这两个函数的调用格式为

F=`ztrans(Fun, `k`, `z`)` 按默认变量进行 Z 变换

F=`iztrans(Fun, `z`, `k`)` Z 反变换，将 z 的函数变换成 k 的函数

若原函数只有一个变量，则调用时无需给出 k 和 z。

【例 5-18】 求解 $f(kT)=akT-2+(akT+2)\mathrm{e}^{-akT}$ 函数的 Z 变换问题。

【求解】 原函数的 Z 变换可以用下面的语句来完成。

```
>> syms a T k  % 声明符号变量
   f=a*k*T-2+(a*k*T+2)*exp(-a*k*T);  % 定义离散函数
   F=ztrans(f); latex(F)   % 求解 Z 变换，并得出其 LaTeX 显示结果
```

该结果可以直接由 LaTeX 排版程序显示如下：

$$\mathscr{Z}[f(kT)]=\frac{aTz}{(z-1)^2}-2\frac{z}{z-1}+\frac{aTz\mathrm{e}^{-aT}}{(z-\mathrm{e}^{-aT})^2}+2\,z\mathrm{e}^{aT}\left(\frac{z}{\mathrm{e}^{-aT}}-1\right)^{-1}$$

【例 5-19】 考虑 $F(z)=q/(z^{-1}-p)^m$ 函数的 Z 反变换问题，这里可以对不同的 m 值进行反变换，并总结出一般规律。

【求解】 根据要求，可以用符号运算工具箱求出 $m=1,2,\cdots,8$ 的 Z 反变换。

```
>> syms p q z
   for i=1:8
      disp(simple(iztrans(q/(1/z-p)^i)))
```

```
   end
 -q/p*(1/p)^n
  q/p^2*(1+n)*(1/p)^n
 -1/2*q*(1/p)^n*(1+n)*(2+n)/p^3
  1/6*q*(1/p)^n*(3+n)*(2+n)*(1+n)/p^4
 -1/24*q*(1/p)^n*(4+n)*(3+n)*(2+n)*(1+n)/p^5
  1/120*q*(1/p)^n*(5+n)*(4+n)*(3+n)*(2+n)*(1+n)/p^6
 -1/720*q*(1/p)^n*(6+n)*(5+n)*(4+n)*(3+n)*(2+n)*(1+n)/p^7
  1/5040*q*(1/p)^n*(7+n)*(6+n)*(5+n)*(4+n)*(3+n)*(2+n)*(1+n)/p^8
```

总结上述结果的规律，可以写出一般的 Z 反变换结果为

$$\mathscr{Z}^{-1}\left[\frac{q}{(z^{-1}-p)^m}\right]=\frac{(-1)^m q}{(m-1)!\,p^{n+m}}\prod_{i=1}^{m-1}(n+i)$$

5.5 复变函数问题的计算机求解

5.5.1 留数的概念与计算

在介绍留数概念之前，应该先介绍一下复变函数解析的概念。若函数 $f(z)$ 在复平面的区域内各点处均为单值，且其导数为有限值，则称 $f(z)$ 在复平面内为解析的，这样就将单值函数上不解析的点称为奇点。假设 $z=a$ 为 $f(z)$ 函数上的奇点，若存在一个最小整数 m 使得乘积 $(z-a)^m f(z)$ 在 $z=a$ 点处解析，则称 $z=a$ 为 m 重奇点。

若 $z=a$ 为 $f(z)$ 函数的单奇点，则可以计算出函数在该奇点处的留数 (residue) 为

$$\operatorname{Res}\Big[f(z),z=a\Big]=\lim_{z\to a}(z-a)f(z) \tag{5-5-1}$$

若 $z=a$ 为函数 $f(z)$ 的 m 重奇点，则该点的留数为

$$\operatorname{Res}\Big[f(z),z=a\Big]=\lim_{z\to a}\frac{1}{(m-1)!}\frac{\mathrm{d}^{m-1}}{\mathrm{d}z^{m-1}}\Big[f(z)(z-a)^m\Big] \tag{5-5-2}$$

求取这样的留数方法很简单，假设已知奇点 a 和重数 m，则用下面的 MATLAB 语句自然可以求出相应的留数。

`c=limit(F*(z-a),z,a)`	单奇点
`c=limit(diff(F*(z-a)^m,z,m-1)/prod(1:m-1),z,a)`	m 重奇点

【例 5-20】试求出函数 $f(z)=\dfrac{1}{z^3(z-1)}\sin\left(z+\dfrac{\pi}{3}\right)\mathrm{e}^{-2z}$ 的留数。

【求解】对原函数的分析可见，$z=0$ 是三重奇点，$z=1$ 是单奇点，故可以直接使用下面的 MATLAB 语句将这两个奇点处的留数分别求出。

```
>> syms z
   f=sin(z+pi/3)*exp(-2*z)/(z^3*(z-1))
```

```
   limit(diff(f*z^3,z,2)/prod(1:2),z,0)
ans =
   -1/4*3^(1/2)+1/2
>> limit(f*(z-1),z,1)
ans =
   1/2*exp(-2)*sin(1)+1/2*exp(-2)*cos(1)*3^(1/2)
```

【例 5-21】 试求函数 $f(z)=\dfrac{\sin z-z}{z^6}$ 的留数。

【求解】 乍看该函数很容易认定 $z=0$ 为 6 重奇点，所以用下面的语句很容易就可以求出该点处的留数值。

```
>> syms z; f=(sin(z)-z)/z^6;
   limit(diff(f*z^6,z,5)/prod(1:5),z,0)
ans =
   1/120
```

其实这里说 $z=0$ 为 6 重奇点有些太保守，不严格地说，从 $k=1$ 开始尝试，能够使得

$$\lim_{z\to 0}\frac{\mathrm{d}^{k-1}}{\mathrm{d}z^{k-1}}z^k f(z)<\infty$$

成立的 k 就是最小的 m，可以考虑 $k=2$，可见导数为无穷大，所以再试验更大的 k 值。对此例子来说，m 的最小值为 $m=3$，留数的值仍然为 1/120。试凑更大的 k 值，如 $k=20$，也不会改变求出的留数值。

```
>> syms z; f=(sin(z)-z)/z^6;
   limit(diff(f*z^2,z,1)/prod(1:1),z,0)
ans =
   Inf
>> limit(diff(f*z^3,z,2)/prod(1:2),z,0) % 再增加阶次
ans =
   1/120
>> limit(diff(f*z^20,z,19)/prod(1:19),z,0) % 再进一步增加阶次
ans =
   1/120
```

可见，若选择的 n 值大于或等于奇点的实际重数，则可以正确得到该函数的留数。在一般应用时可选择一个较大的 n 值来求取留数。

【例 5-22】 试求出 $f(z)=\dfrac{1}{z\sin z}$ 函数的留数。

【求解】 分析该函数，因为 $\sin z$ 在 $z=0$ 点的收敛速度和 z 是一样的，显然，$z=0$ 点为 $f(z)$ 的二重奇点，这时，相应的留数可以用下面语句求出。

```
>> syms z; f=1/(z*sin(z));
   c0=limit(f*z^2,z,0)
c0 =
```

```
     1
```

进一步分析给定函数 $f(z)$，可以发现该函数在 $z=\pm k\pi$ 处均不解析，其中 k 为正整数，且这些点是原函数的单奇点，由于 MATLAB 的符号运算工具箱并未给出整数的定义，所以这里只能对一些 k 值进行试探，求出它们的留数，最后将结果归纳成所需的公式。

```
>> k=[-4 4 -3 3 -2 2 -1 1]; c=[];
   for kk=k; c=[c,limit(f*(z-kk*pi),z,kk*pi)]; end; c
c =
   [ -1/4/pi,  1/4/pi,  1/3/pi, -1/3/pi, -1/2/pi,  1/2/pi,   1/pi,   -1/pi]
```

综上，可以归纳出 $\mathrm{Res}[f(z),z=\pm k\pi]=\pm(-1)^k\dfrac{1}{k\pi}$。

5.5.2 有理函数的部分分式展开

考虑有理函数

$$G(x)=\frac{B(x)}{A(x)}=\frac{b_1x^m+b_2x^{m-1}+\cdots+b_mx+b_{m+1}}{x^n+a_1x^{n-1}+a_2x^{n-2}+\cdots+a_{n-1}x+a_n} \tag{5-5-3}$$

其中，a_i 和 b_i 均为常数。有理函数的互质概念是一非常重要的概念。所谓互质，就是指多项式 $A(x)$ 和 $B(x)$ 没有公约数。对一般给定的两个多项式来说，用手工方式判定多项式互质还是比较困难的，但可以利用 MATLAB 符号运算工具箱中的 `gcd()` 函数可以直接求出两个多项式的最大公约数。该函数的调用方法为

C=gcd(*A*,*B*)

其中，A 和 B 分别表示两个多项式，该函数将得出这两个多项式的最大公约数 C，若得出的 C 为多项式，则两个多项式为非互质的多项式，这时两个多项式可用约简为 A/C 和 B/C。

【例 5-23】给出两个多项式

$$\begin{aligned}A(x)&=x^4+7x^3+13x^2+19x+20\\B(x)&=x^7+16x^6+103x^5+346x^4+655x^3+700x^2+393x+90\end{aligned}$$

试判定它们是否互质。

【求解】求解这样的问题可用采用 MATLAB 语言提供的 `gcd()` 函数完成。

```
>> syms x; A=x^4+7*x^3+13*x^2+19*x+20;
   B=x^7+16*x^6+103*x^5+346*x^4+655*x^3+700*x^2+393*x+90;
   d=gcd(A,B)
d =
   x+5
```

可见，两个多项式具有最大公约数 $(x+5)$，故两个多项式不是互质的，这两个多项式可用进一步简化为

```
>> simple(A/d)
```

```
ans =
   x^3+2*x^2+3*x+4
>> simple(B/d)
ans =
  (x+2)*(x+3)^2*(x+1)^3
```

若互质多项式 $A(x)=0$ 的根均为相异的值 $-p_i, i=1,2,\cdots,n$，则可以将 $G(x)$ 函数写成下面的部分分式展开形式。

$$G(x)=\frac{r_1}{x+p_1}+\frac{r_2}{x+p_2}+\cdots+\frac{r_n}{x+p_n} \tag{5-5-4}$$

其中 r_i 称为留数，简记作 $\mathrm{Res}[G(-p_i)]$，其值可以由下面的极限式求出。

$$r_i=\mathrm{Res}[G(-p_i)]=\lim_{x\to -p_i} G(s)(x+p_i) \tag{5-5-5}$$

如果分母多项式中含有 $(x+p_i)^k$ 项，亦即 $-p_i$ 为 m 重根，则相对这部分特征根的部分分式展开项可以写成

$$\frac{r_i}{x+p_i}+\frac{r_{i+1}}{(x+p_i)^2}+\cdots+\frac{r_{i+k-1}}{(x+p_i)^k} \tag{5-5-6}$$

这时，r_{i+j-1} 可以用下面的公式直接求出。

$$r_{i+j-1}=\frac{1}{(k-1)!}\lim_{x\to -p_i}\frac{\mathrm{d}^{j-1}}{\mathrm{d}x^{j-1}}\left[G(x)(x+p_i)^k\right],\ j=1,2,\cdots,k \tag{5-5-7}$$

MATLAB 语言中给出了现成的数值函数 `residue()` 求取有理函数 $G(x)$ 的部分分式展开表示，该函数的调用格式为

```
[r, p, k]=residue(b, a)
```

其中，$\boldsymbol{a}=[1,a_1,a_2,\cdots,a_n]$，$\boldsymbol{b}=[b_1,b_2,\cdots,b_m]$，返回的 $\boldsymbol{r}$ 和 $\boldsymbol{p}$ 向量为式 (5-5-4) 的 r_i 系数，若有重根则应该相应地由式 (5-5-6) 中给出的系数取代。k 为余项，对 $m<n$ 的函数来说该项为空矩阵。该函数并未给出 $-p_i$ 是否为重根的自动判定功能，所以部分分式展开的结果需要手动写出。

【例 5-24】试求下面有理函数的部分分式展开。

$$G(s)=\frac{s^3+2s^2+3s+4}{s^6+11s^5+48s^4+106s^3+125s^2+75s+18}$$

【求解】用下面的语句可以求出该函数的部分分式展开为

```
>> n=[1,2,3,4]; d=[1,11,48,106,125,75,18]; format long
   [r,p,k]=residue(n,d); [n,d1]=rat(r); [n,d1,p]
ans =
 -17.0000000000000   8.0000000000000  -2.99999999999995
```

```
 -7.00000000000000   4.00000000000000  -2.99999999999995
  2.00000000000000   1.00000000000000  -2.00000000000016
  1.00000000000000   8.00000000000000  -0.99999999999998
 -1.00000000000000   2.00000000000000  -0.99999999999998
  1.00000000000000   2.00000000000000  -0.99999999999998
```

其中，p 为奇点向量，n, d_1 为每个 p 值对应系数的分子和分母数值。由数值方法直接求出的分母多项式的根为小数，有一些误差。事实上，该分母多项式的特征根为：-3 为二重奇点，-2 为单奇点，-1 为三重奇点。分析奇点的情况，可以写出部分分式展开为

$$G(s) = -\frac{17}{8(s+3)} - \frac{7}{4(s+3)^2} + \frac{2}{s+2} + \frac{1}{8(s+1)} - \frac{1}{2(s+1)^2} + \frac{1}{2(s+1)^3}$$

【例 5-25】写出下面式子的部分分式展开。

$$G(s) = \frac{2s^7+2s^3+8}{s^8+30s^7+386s^6+2772s^5+12093s^4+32598s^3+52520s^2+45600s+16000}$$

【求解】采用 MATLAB 自带的 residue() 函数，则只能求解得出数值解，对本例给出的问题，可以用下面的语句直接求出有理函数的部分分式展开式子为

```
>> n=[2,0,0,0,2,0,0,8];
   d=[1,30,386,2772,12093,32598,52520,45600,16000];
   [r,p]=residue(n,d)
 ans =
1.0e+004 *
    4.99959030930686  -0.00050000000003
    2.84885832580441  -0.00050000000003
    1.30409999762507  -0.00050000000003
   -5.04731527861460  -0.00039999999997
    2.14495555022347  -0.00039999999997
   -0.54818333201362  -0.00039999999997
    0.00012222222224  -0.00020000000000
    0.00000023148148  -0.00010000000000
```

从得出的结果较难判定重根情况，故难以准确写出部分分式展开式子。由得出的数据，假定 $p_1=-5$ 为三重实根，$p_2=-4$ 为三重实根，$p_3=-2$, $p_4=-1$ 为单个实根，故可以写出部分分式展开的表达式为

$$\frac{49995.9030930686}{(s+5)} + \frac{28488.5832580441}{(s+5)^2} + \frac{13040.9999762507}{(s+5)^3} - \frac{50473.1527861460}{(s+4)} + \frac{21449.5555022347}{(s+4)^2} - \frac{5481.3333201362}{(s+4)^3} + \frac{1.2222222224}{(s+2)} + \frac{0.0023148148}{(s+1)}$$

应该指出，这样的展开方式在分母上作了近似，另外这些特征根假定为重根也是一种令人不大放心的假设，所以对这样的问题，可以考虑编写更好的解析算法。利用

MATLAB 符号运算工具箱中定义的精确求根功能和式 (5-5-5) 与 (5-5-7) 的留数计算公式，可以立即编写出如下的扩展函数 residue1()。该函数仍应该置于 @sym 目录下，其清单为

```
function f=residue1(F,s)
f=sym(0); if nargin==1, syms s; end
[num,den]=numden(F);
x=solve(den); [x0,ii]=sort(double(x)); x=[x(ii);rand(1)];
k_vec=find(diff(double(x))~=0); ee=x(k_vec);
k_vec=[k_vec(1); diff(k_vec(:,1))];
for i=1:length(k_vec),
   for j=1:k_vec(i), m=k_vec(i); s0=ee(i);
      k=limit(diff(F*(s-s0)^m,s,j-1),s,s0);
      f=f+k/(s-s0)^(m-j+1)/factorial(j-1);
 end, end
```

该函数的调用格式为

f=residue1(F,s)

其中，F 为有理函数的解析表达式，s 为自变量。返回的结果 f 是部分分式展开的表达式。

【例 5-26】考虑例 5-24 中给出的函数 $f(s)$，用解析方式求出其部分分式展开。

【求解】用下面的语句可立即得出该函数的部分分式展开式，如下所示，该结果与原例中数值结果完全一致。

```
>> syms s;
   G=(s^3+2*s^2+3*s+4)/(s^6+11*s^5+48*s^4+106*s^3+125*s^2+75*s+18);
   G1=residue1(G,s)
G1 =
    -7/4/(x+3)^2-17/8/(x+3)+2/(x+2)+1/2/(x+1)^3-1/2/(x+1)^2+1/8/(x+1)
```

【例 5-27】仍考虑例 5-25 中给出的有理函数 $G(s)$，试用解析方法写出其部分分式展开。

【求解】由于前面例子中使用的方法是数值方法，所以未必很准确。利用新编写的 residue1() 函数，则可以用下面的语句对之进行部分分式展开。

```
>> syms s
   G=(2*s^7+2*s^3+8)/...   % 其中 ... 表示续行
    (s^8+30*s^7+386*s^6+2772*s^5+12093*s^4+32598*s^3+52520*s^2+45600*s+16000);
   f=residue1(G); latex(f)
```

这样，可以得出原分式的部分分式展开为

$$\frac{13041}{(s+5)^3}+\frac{341863}{12(s+5)^2}+\frac{7198933}{144(s+5)}-\frac{16444}{3(s+4)^3}+\frac{193046}{9(s+4)^2}-\frac{1349779}{27(s+4)}+\frac{11}{9(s+2)}+\frac{1}{432(s+1)}$$

若将得出的部分分式展开减去原函数并化简，则结果为 0，表示得出的结果是正确的。

```
>> simple(f-G)
ans =
   0
```

对比例 5-25 中给出的结果，这时得出的结果更令人信服。

【例 5-28】考虑例 5-23 中的非互质多项式构成的有理函数 $G(x) = A(x)/B(x)$，试用数值方法和解析方法写出其部分分式展开。

【求解】用数值方法进行部分分式展开将得出如下的结果。

```
>> syms x; A=x^4+7*x^3+13*x^2+19*x+20;
   B=x^7+16*x^6+103*x^5+346*x^4+655*x^3+700*x^2+393*x+90;
   n=sym2poly(A); d=sym2poly(B);
   [r,p,k]=residue(n,d); [n1,d1]=rat(r); [n1,d1,p]
ans =
                     0   1.00000000000000  -4.99999999999977
  -17.00000000000000   8.00000000000000  -3.00000000000015
   -7.00000000000000   4.00000000000000  -3.00000000000015
    2.00000000000000   1.00000000000000  -1.99999999999976
    1.00000000000000   8.00000000000000  -1.00000000000005
   -1.00000000000000   2.00000000000000  -1.00000000000005
    1.00000000000000   2.00000000000000  -1.00000000000005
```

然而，r_1 的值为一个很小的数值，由数值方法不能完全得出 0，所以数值方法将产生误差。

```
>> r(1)
ans =
   -2.605785957382269e-014
```

用前面介绍解析解方法可用得出和 $x = -5$ 奇点完全无关的解析解，亦即 residue1() 函数同样适合于非互质有理函数的部分分式展开。

```
>> residue1(A/B,x)
ans =
   -7/4/(x+3)^2-17/8/(x+3)+2/(x+2)+1/2/(x+1)^3-1/2/(x+1)^2+1/8/(x+1)
```

【例 5-29】求有理函数的部分分式展开。

$$G(x) = \frac{-17x^5 - 7x^4 + 2x^3 + x^2 - x + 1}{x^6 + 11x^5 + 48x^4 + 106x^3 + 125x^2 + 75x + 17}$$

【求解】可以先试用 residue() 函数。

```
>> syms x;  G=(-17*x^5-7*x^4+2*x^3+x^2-x+1)...
        /(x^6+11*x^5+48*x^4+106*x^3+125*x^2+75*x+17);
   G1=latex(residue1(G,x)),
G1 =
    0.
```

显然，得出的部分分式展开式是错误的。

前面编写的 residue1() 函数一个已知的错误，该函数要求有理函数分母多项式方程 $D(x)=0$ 的解必须能精确求出，否则不能进行正确转换。这是因为采用式 (5-5-2) 求取留数算法的原因。如果原有理函数分母 $D(x)$ 的某无理数根为 x_0，通过任何的求根算法根本不能在有限位内得出 x_0 的精确值，只能得到其近似值 $\hat{x}_0$，所以代入式 (5-5-2) 则有

$$\operatorname{Res}[f(\hat{x}_0)]=\lim_{x\to\hat{x}_0}\frac{1}{(m-1)!}\frac{\mathrm{d}^{m-1}}{\mathrm{d}x^{m-1}}\left[f(\hat{x}_0)(x-\hat{x}_0)^m\right] \tag{5-5-8}$$

这时，因为 $\hat{x}_0$ 不是原方程的根，所以 $f(\hat{x}_0)$ 为有限数值，故整个极限等于 0，从而得出错误的结论，该项的留数等于 0。

基于此问题，可以考虑对原算法加以改进。假设用 vpa() 函数能求出多项式方程所有的根 $x_i, i=1,2,\cdots,n$，可以由这些根重组多项式的分母而取代原来的分母，则能得出新的函数 $f_1(x)$ 来取代式 (5-5-8) 中的 $f(x)$，这样就能确保 $f_1(x)$ 和 $(x-\hat{x}_0)^m$ 在 $\hat{x}_0$ 处能真正相销，得出原函数的留数。修改后的函数内容为

```
function f=residue(F,s)
f=sym(0); if nargin==1, syms s; end
[num,den]=numden(F); x0=solve(den);
[x,ii]=sort(double(x0)); x0=x0(ii); x=[x0;rand(1)];
kvec=find(diff(double(x))~=0); ee=x(kvec);
kvec=[kvec(1); diff(kvec(:,1))];
a0=limit(den/s^length(x0),s,inf); % 求分母首项系数
F1=num/(a0*prod(s-x0));    % 重组 f(x) 函数的分母，得出新函数 f1(x)
for i=1:length(kvec),
   for j=1:kvec(i), m=kvec(i); s0=ee(i);
      k=limit(diff(F1*(s-s0)^m,s,j-1),s,s0);   % 这里用 f1(x)
      f=f+k/(s-s0)^(m-j+1)/factorial(j-1);
   end
end
```

【例 5-30】试用新编写的 residue() 函数求出例 5-29 中有理函数的部分分式展开。

【求解】用前面的 residue1() 函数得出的部分分式展开式子是错误的。采用 residue() 函数将得出如下结果：

```
>> G1=latex(residue(G,x)),
```

其展开结果如下

$$\frac{0.2125567969626322944185008686013 4}{x+0.5208596052932005211731808946575 7}-\frac{556.2565306869610416947605966344 5}{x+3.2617310107385187021102903315402}+$$

$$\frac{0.8794649261946206598661076890531 3+\mathrm{j}5.4970762578573277245912249310211}{x+1.0777588719352924223318783254230+\mathrm{j}0.6021065910607614855934530950638 9}+$$

$$\frac{0.8794649261946206598661076890531 3-\mathrm{j}5.4970762578573277245912249310211}{x+1.0777588719352924223318783254230-\mathrm{j}0.6021065910607614855934530950638 9}+$$

$$\frac{268.6425220188045840403049401938\,7-\mathrm{j}349.1231094995268181442271489307\,9}{x+2.5309458200488479660263860614781+\mathrm{j}0.39976310544985541814054726451926}+$$
$$\frac{268.6425220188045840403049401938\,7+\mathrm{j}349.1231094995268181442271489307\,9}{x+2.5309458200488479660263860614781-\mathrm{j}0.39976310544985541814054726451926}$$

该算法的精度大大高于 MATLAB 自身的数值 residue() 函数。

5.5.3 基于部分分式展开的 Laplace 变换

符号运算工具箱中提供的 Laplace 反变换函数 ilaplace() 可以较好地解决一般函数的 Laplace 反变换问题。但一类问题，例如带有复特征根的有理函数的 Laplace 反变换问题不适合由该函数直接求解，这已经在例 5-3 中说明了，直接用该函数求解出的解可读性很差。

观察例 5-30 的结果可以立即发现，若某项部分分式展开式子为 $(a+\mathrm{j}b)/(s+c+\mathrm{j}d)$，则一定会含有其共轭项 $(a-\mathrm{j}b)/(s+c-\mathrm{j}d)$，这样就需要对下面的式子进行化简。

$$(a+\mathrm{j}b)\mathrm{e}^{(c+\mathrm{j}d)t}+(a-\mathrm{j}b)\mathrm{e}^{(c-\mathrm{j}d)t}=\alpha\mathrm{e}^{ct}\sin(dt+\phi) \tag{5-5-9}$$

其中，$\alpha=-2\sqrt{a^2+b^2}$，且 $\phi=-\tan^{-1}(b/a)$。

有了这样的算法，则可以用 MATLAB 语言编写出实现上述算法的数值函数 pfrac()。该函数基于 MATLAB 的原始 residue() 函数，其内容为

```
function [R,P,K]=pfrac(num,den)
[R,P,K]=residue(num,den);
for i=1:length(R),
   if imag(P(i))>eps
      a=real(R(i)); b=imag(R(i));
      R(i)=-2*sqrt(a^2+b^2); R(i+1)=-atan2(a,b);
   elseif abs(imag(P(i)))<eps
      R(i)=real(R(i));
end, end
```

该函数的调用格式为

[r,p,K]=pfrac(num,den)

其中，p 和 K 的定义和 residue() 函数一致，r 稍有不同，若相应的 p_i 项为实数，则 r_i 和 residue() 也一致，若某个 p_i 的值为复数，则 r_i, r_{i+1} 项分别为相应的 α 和 ϕ 值。

【例 5-31】重新考虑例 5-30 中的问题，可以用 MATLAB 数值算法得出如下结果：

```
>> num=[-17,-7,2,1,-1,1]; den=[1,11,48,106,125,75,17];
   [r,p,k]=pfrac(num,den); format long e; [r,p]
ans =
  -5.562565306867494e+002   -3.261731010738623e+000
   2.125567969626306e-001   -5.208596052932000e-001
```

```
-8.810351870908827e+002     -2.530945820048808e+000 +3.997631054499472e-001i
-6.558508770732739e-001     -2.530945820048808e+000 -3.997631054499472e-001i
-1.113396711708849e+001     -1.077758871935285e+000 +6.021065910607617e-001i
-2.982949324280423e+000     -1.077758871935285e+000 -6.021065910607617e-001i
```

这样，可以写出有理函数的可读性更好的 Laplace 反变换为

$$\begin{aligned}\mathscr{L}^{-1}[F(s)] = &-556.2565306867494\mathrm{e}^{-3.261731010738623t} + 2.125567969626306\mathrm{e}^{-0.5208596052932t}\\ &-881.0351870908827\mathrm{e}^{-2.530945820048808t}\sin(0.3997631054499472t - 0.6558508770732739)\\ &-11.13396711708849\mathrm{e}^{-1.077758871935285t}\sin(0.6021065910607617t - 2.982949324280423)\end{aligned}$$

比较此结果与例 5-3 中的结果，显而易见这个结果更简洁。

5.5.4 封闭曲线积分问题计算

考虑如下定义的曲线积分

$$\oint_\Gamma f(z)\mathrm{d}z \tag{5-5-10}$$

其中，Γ 为二维平面内的走行方向为逆时针的封闭曲线，如果积分线为顺时针的，则应该将被积函数乘以 -1。这时假设该封闭曲线内包围 m 个奇点，p_i，$(i=1,2,\cdots,m)$，则可以分别用前面的算法求出这些奇点上的留数为 $\mathrm{Res}[f(p_i)]$，这时封闭曲线积分的值等于 $2\pi\mathrm{j}$ 乘以这些留数的和，即

$$\oint_\Gamma f(z)\mathrm{d}z = 2\pi\mathrm{j}\sum_{i=1}^{m}\mathrm{Res}[f(p_i)] \tag{5-5-11}$$

如果曲线的走行方向是顺时针的，则可以将得出的结果反号。

【例 5-32】试求 $f(z)$ 函数在 $|z|=6$ 曲线上的曲线积分。其中，$f(z)$ 函数为

$$f(z) = \frac{2z^7+2z^3+8}{z^8+30z^7+386z^6+2772z^5+12093z^4+32598z^3+52520z^2+45600z+16000}$$

【求解】可以用例 5-27 中给出的方法求出其部分分式展开为

$$\frac{13041}{(s+5)^3}+\frac{341863}{12(s+5)^2}+\frac{7198933}{144(s+5)}-\frac{16444}{3(s+4)^3}+\frac{193046}{9(s+4)^2}-\frac{1349779}{27(s+4)}+\frac{11}{9(s+2)}+\frac{1}{432(s+1)}$$

可见，该函数的奇点为：$p_1=-1$ 为单奇点，$p_2=-2$ 亦为单奇点，$p_3=-4, p_4=-5$ 均为 3 重奇点。由上面的部分分式展开式子可知，各个奇点的留数为部分分式展开的一次项的分子值，这样可以得出所需的曲线积分值为

$$\oint_{|z|=6} f(z)\mathrm{d}z = 2\pi\mathrm{j}\left[\frac{1798933}{144}-\frac{1349779}{27}+\frac{11}{9}+\frac{1}{432}\right] = 4\pi\mathrm{j}$$

【例 5-33】试求出下面的曲线积分[50]

$$\oint_\Gamma \frac{1}{(z+\mathrm{j})^{10}(z-1)(z-3)}\mathrm{d}z$$

其中，Γ 为 $|z|=2$ 的逆时针圆周。

【求解】 经过简单观察原函数，可以发现该函数在 $z=1, z=3$ 处有单个奇点，在 $z=-\mathrm{j}$ 处有 10 重奇点，又因为给定的 Γ 为 $|z|=2$ 圆周，所以 $z=1, z=-\mathrm{j}$ 在该圆周的范围内，$z=3$ 在该区域外，所以不必计算该留数，所以原曲线积分的值可以用下面的语句直接求出。

```
>> i=sym(sqrt(-1)); syms z
   f=1/((z+i)^10*(z-1)*(z-3));   % 定义被积函数
   r1=limit(diff(f*(z+i)^10,z,9)/prod(1:9),z,-i);
   r2=limit(f*(z-1),z,1); a=2*pi*i*(r1+r2)
a =
   (237/312500000+779/78125000*i)*pi
```

根据文献 [50] 中给出的方法，手工求解时建议先计算 $z=3$ 的留数，故得出整个环路积分值为 $-\pi\mathrm{j}/(3+\mathrm{j})^{10}$。下面可以证明二者是完全一致的。

```
>> a+pi*i/(3+i)^10
ans =
   0
```

若曲线 Γ 的方程为 $|z|=4$，则该曲线将 $z=1,3,\mathrm{j}$ 三个奇点均包围在内，这时曲线积分应该和这三个留数的和有关，故可以用下面的语句求出曲线积分的值为

```
>> r3=limit(f*(z-3),z,3); b=2*pi*i*(r1+r2+r3)
b =
   0
```

既然用符号运算的方法计算留数特别简单，所以没有必要在用间接的方法计算了，可以通过式 (5-5-11) 中给出的算法直接求出。

利用变量替换的方法可以将开区间积分变换成封闭曲线积分。例如，有的书上用变换的方法求解 $f(x)=\sin x/x$ 在 $(0,\infty)$ 区域的无穷积分。而事实上，这样的积分直接用 MATLAB 中的 `int()` 函数就可以求解，所以本书不介绍这类方法。

5.6 本章要点简介

- 本章有关的 MATLAB 函数及自编函数由下表给出。

函数名	函 数 功 能	工具箱	本书页码
`laplace()`	函数的 Laplace 变换	符号运算	143
`ilaplace()`	函数的 Laplace 反变换	符号运算	143
`fourier()`	函数的 Fourier 变换	符号运算	147
`ifourier()`	函数的 Fourier 反变换	符号运算	147
`fouriersin`	函数的 Fourier 正弦变换，还可以通过符号积分求解	Maple	150
`fouriercos`	函数的 Fourier 余弦变换	Maple	150
`invfouriersin`	函数的 Fourier 正弦反变换	Maple	150

续表

函数名	函 数 功 能	工具箱	本书页码
`mellin`	函数的 Mellin 变换	Maple	153
`invmellin`	函数的 Mellin 反变换	Maple	153
`hankel`	函数的 Hankel 变换	Maple	154
`invhankel`	函数的 Hankel 反变换	Maple	154
`ztrans()`	函数的 Z 变换	符号运算	155
`iztrans()`	函数的 Z 反变换	符号运算	155
语句	求取函数单奇点和重奇点的留数	自编	156
`gcd()`	函数的最大公约数，`lcd()` 可以求最小公倍数	符号运算	158
`residue()`	有理函数的部分分式展开，数值方法	MATLAB	159
`residue1()`	利用符号运算的有理函数的部分分式展开	自编	161
`residue()`	利用符号运算的有理函数的部分分式展开 (改进版)	自编	163

- Laplace 变换是一种很重要的积分变换方法。本章中介绍了 Laplace 变换及其反变换的定义和性质，并着重介绍了正反 Laplace 变换的 MATLAB 求解方法。

- Fourier 变换是另一类常用的积分变换方法，可以用于信号的频域分析。本章介绍了 Fourier 正反变换的定义和性质，介绍了利用 MATLAB 语言求解 Fourier 变换的方法，还探讨了几种特殊的 Fourier 变换及 MATLAB 求解方法，如正弦、余弦 Fourier 变换、离散 Fourier 正余弦变换等，并介绍了直接积分方法和用 MATLAB 调用 Maple 语言现成变换函数的方法。

- 本章还介绍了两种不太常用的积分变换 —— Mellin 变换和 Hankel 变换，这些变换在 MATLAB 的符号运算工具箱中没有直接对应的函数，但可以通过符号运算工具箱中积分功能直接计算，也可以从 MATLAB 调用 Maple 语言的相应函数计算。本章同时还指出了这些正反变换的有些函数是不难用计算机数学语言求解的，只能通过手工推导或查阅数学手册[41]的方式求解。

- 在离散系统研究中一类很重要的变换 —— Z 变换的定义和性质等也在本章中给出，在此基础上着重介绍了 Z 变换及其反变换的 MATLAB 求解方法，并通过例子演示了各类函数 Z 变换的实际求解。

- 复变函数中一类重要的问题是奇点与留数的概念与求取。本章介绍了利用 MATLAB 语言进行留数运算的方法，并以其为基础编写了有理函数部分分式展开的求解函数，比 MATLAB 自己提供的 `residue()` 函数更进了一步。本章还讨论了有理函数 Laplace 反变换的求解方法和化简方法，基于留数定理还探讨了封闭曲线积分的求解方法。

5.7 习题

1 对下列的函数 $f(t)$ 进行 Laplace 变换。

① $f_a(t)=\dfrac{\sin\alpha t}{t}$, ② $f_b(t)=t^5\sin\alpha t$, ③ $f_c(t)=t^8\cos\alpha t$ ④ $f_d(t)=t^6\mathrm{e}^{\alpha t}$

⑤ $f_e(t)=5\mathrm{e}^{-at}+t^4\mathrm{e}^{-at}+8\mathrm{e}^{-2t}$, ⑥ $f_f(t)=\mathrm{e}^{\beta t}\sin(\alpha t+\theta)$, ⑦ $f_g(t)=\mathrm{e}^{-12t}+6\mathrm{e}^{9t}$

2 对下面的 $F(s)$ 式进行 Laplace 反变换。

① $F_a(s)=\dfrac{1}{\sqrt{s^2(s^2-a^2)(s+b)}}$, ② $F_b(s)=\sqrt{s-a}-\sqrt{s-b}$, ③ $F_c(s)=\ln\dfrac{s-a}{s-b}$

④ $F_d(s)=\dfrac{1}{\sqrt{s}(s+a)}$, ⑤ $F_e(s)=\dfrac{3a^2}{s^3+a^3}$, ⑥ $F_f(s)=\dfrac{(s-1)^8}{s^7}$

⑦ $F_g(s)=\ln\dfrac{s^2+a^2}{s^2+b^2}$, ⑧ $F_h(s)=\dfrac{s^2+3s+8}{\prod_{i=1}^{8}(s+i)}$, ⑨ $F_i(s)=\dfrac{1}{2}\dfrac{s+\alpha}{s-\alpha}$

3 Laplace 变换的一个重要应用是求解常系数线性微分方程，可以利用当函数具有零初始值和各阶导数的零初始值下 $\mathscr{L}[\mathrm{d}^n f(t)/\mathrm{d}t^n]=s^n\mathscr{L}[f(t)]$ 这一性质对微分方程进行 Laplace 变换的方法其求解微分方程。试使用这样的方法求解下面的微分方程。

$y''(t)+3y'(t)+2y(t)=\mathrm{e}^{-t}$, $y(0)=y'(0)=0$

4 试求出下面函数的 Fourier 变换，对得出的结果再进行 Fourier 反变换，观察是否能得出原来函数。

① $f(x)=x^2(3\pi-2|x|)$, $0\leqslant x\leqslant 2\pi$, ② $f(t)=t^2(t-2\pi)^2$, $0\leqslant t\leqslant 2\pi$

③ $f(t)=\mathrm{e}^{-t^2}$, $-l\leqslant t\leqslant l$, ④ $f(t)=t\mathrm{e}^{-|t|}$, $-\pi\leqslant t\leqslant\pi$

5 试证明 $\cos\theta+\cos 2\theta+\cdots+\cos n\theta=\dfrac{\sin(n\theta/2)\cos[(n+1)\theta/2]}{\sin\theta/2}$。

6 试求出下面函数的 Fourier 正弦和余弦变换，并用 Fourier 正弦、余弦反变换对得出的结果进行处理，观察是否能还原成原始函数。

① $f(t)=\mathrm{e}^{-t}\ln t$, ② $f(x)=\dfrac{\cos x^2}{x}$, ③ $f(x)=\ln\dfrac{1}{\sqrt{1+x^2}}$

7 试求下面函数的离散 Fourier 正弦、余弦变换。

① $f(x)=\mathrm{e}^{kx}$ ② $f(x)=x^3$

8 试对下面的函数进行 Mellin 变换。

$$f(x)=\begin{cases}\sin(a\ln x), & x\leqslant 1\\ 0, & \text{其他 } x\end{cases}$$

9 请将下述时域序列函数 $f(kT)$ 进行 Z 变换，并对结果进行反变换检验。

① $f_a(kT)=\cos(kaT)$, ② $f_b(kT)=(kT)^2\mathrm{e}^{-akT}$, ③ $f_c(kT)=\dfrac{1}{a}(akT-1+\mathrm{e}^{-akT})$

④ $f_d(kT)=\mathrm{e}^{-akT}-\mathrm{e}^{-bkT}$, ⑤ $f_e(kT)=\sin(\alpha kT)$, ⑥ $f_f(kT)=1-\mathrm{e}^{-akT}(1+akT)$

10 已知下述各个 Z 变换表达式 $F(z)$，试对它们分别进行 Z 反变换。

① $F_a(z)=\dfrac{10z}{(z-1)(z-2)}$，② $F_b(z)=\dfrac{z^2}{(z-0.8)(z-0.1)}$，③ $F_c(z)=\dfrac{z}{(z-a)(z-1)^2}$

④ $F_d(z)=\dfrac{z^{-1}(1-\mathrm{e}^{-aT})}{(1-z^{-1})(1-z^{-1}\mathrm{e}^{-aT})}$，⑤ $F_e(z)=\dfrac{Az[z\cos\beta-cos(\alpha T-\beta)]}{z^2-2z\cos(\alpha T)+1}$

11 已知某信号的 Laplace 变换为 $\dfrac{b}{s^2(s+a)}$，试求其 Z 变换，并验证结果。

12 试用计算机证明

$$\mathscr{Z}\left\{1-\mathrm{e}^{-akT}\left[\cos(bkT)+\frac{a}{b}\sin(bkT)\right]\right\}=\frac{z(Az+B)}{(z-1)(z^2-2\mathrm{e}^{-aT}\cos(bT)z+\mathrm{e}^{-2aT})}$$

式中，$A=1-\mathrm{e}^{-aT}\cos(bT)-\dfrac{a}{b}\mathrm{e}^{-aT}\sin(bT)$，$B=\mathrm{e}^{-2aT}+\dfrac{a}{b}\mathrm{e}^{-aT}\sin(bT)-\mathrm{e}^{-aT}\cos(bT)$。

13 试求出 $f(x)=\dfrac{x^2+4x+3}{x^5+4x^4+3x^3+2x^2+5x+2}\mathrm{e}^{-5x}$ 的奇点、奇点重数及各个奇点处的留数。

14 试求出下面有理函数的部分分式展开，并用 LaTeX 的形式比较好地显示得出的结果。

$$G(s)=\frac{s+5}{s^8+21s^7+181s^6+839s^5+2330s^4+4108s^3+4620s^2+3100s+1000}$$

15 求下面的封闭环路积分。

① $\displaystyle\oint_\Gamma\frac{z^{15}}{(z^2+1)^2(z^4+2)^3}\mathrm{d}z$，其中，$\Gamma$ 为 $|z|=3$ 正向圆周

② $\displaystyle\oint_\Gamma\frac{z^3}{1+z}\mathrm{e}^{1/z}\mathrm{d}z$，其中，$\Gamma$ 为 $|z|=2$ 正向圆周



第 6 章　代数方程与最优化问题的计算机求解

方程求解问题是科学与工程研究中经常遇到的问题。线性代数方程可以用第 4 章中介绍的方法直接求解，非线性方程或一般多项式方程的求解将在第 6.1 节中介绍，在该节中将先介绍一元或二元方程的图解法，然后将介绍基于 MATLAB 符号运算工具箱的多项式方程或一类可以化成多项式方程的代数方程的解析解方法，再介绍一般非线性方程组的数值解法。

最优化技术是当前科学研究中的一类重要的手段。所谓最优化就是找出使得目标函数值达到最小或最大的自变量值的方法。从其分类看有无约束最优化问题和有约束最优化问题。第 6.2 节详细介绍了无约束最优化问题以及 MATLAB 求解方法，介绍了图解法和一般的数值算法，并引入了全局最优解与局部最优解的概念。第 6.3 节介绍了有约束最优化的概念，引入了约束可行区域的概念，并就线性规划问题、二次型规划问题和一般非线性规划问题的 MATLAB 语言求解进行了详细的介绍。第 6.4 节进一步引申了最优化问题，引入了整数规划的概念，并介绍了如何用 MATLAB 语言求解混合整数线性规划问题，并介绍了基于"分枝定界法"的一般整数规划问题及其 MATLAB 实现。通过本章内容的学习，读者应该能掌握一般非线性方程及非线性最优化问题的实际求解方法。

6.1　代数方程的求解

6.1.1　代数方程的图解法

MATLAB 提供了很强的一元、二元隐函数绘制功能，充分利用这些功能就可以将一元、二元的方程用曲线表示，并由曲线的交点读出方程的实数根来。然而，方程的图解法是有局限性的，仅适用于一元、二元方程，多元方程是不能用图解法直接求解的。本节将通过例子演示一元、二元方程的求根问题。

6.1.1.1　一元方程的图解法

用 `ezplot()` 函数可以绘制出给定的隐函数 $f(x)=0$ 曲线，所以可以用图解法从给出的曲线和 $y=0$ 线的交点上读出所有的实数解。

【例 6-1】 用图解法求解方程 $\mathrm{e}^{-3t}\sin(4t+2)+4\mathrm{e}^{-0.5t}\cos(2t)=0.5$。

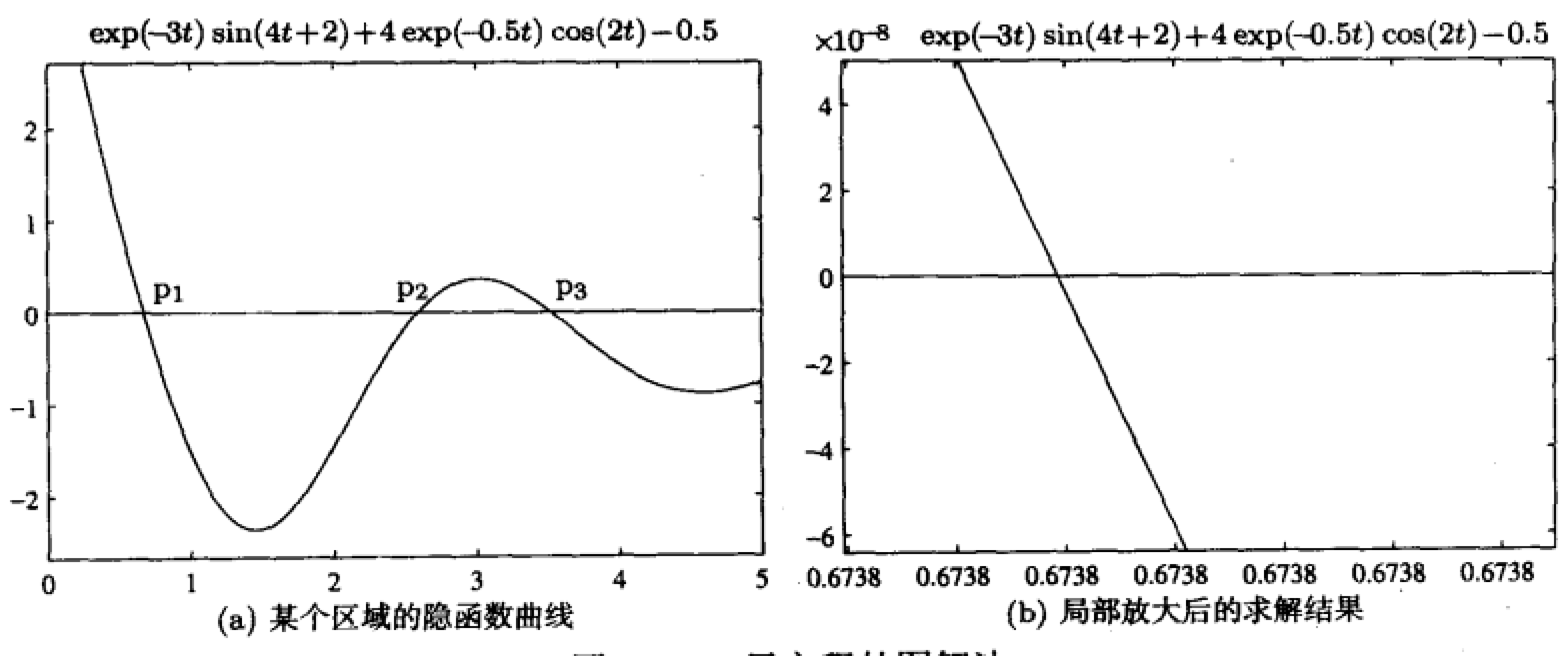
【求解】 用 `ezplot()` 函数可以绘制出如图 6-1 (a) 所示的曲线，该曲线与横轴的所有交点均是原一元方程的解。

```
>> ezplot('exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5',[0 5])
   hold on, line([0,5],[0,0])% 同时绘制横轴
```

从得出曲线可以看出，该方程可能有多个实根，如果想用图解法得出某个根，则可以对该点附近局部放大，直到曲线穿越 0 线，且 t 轴给出的各个标点的数值完全一致时，则可以断定原方

程的解即为 t 轴标度，如图 6-1 (b) 所示，即该方程的一个解为 $t=3.5203$。将其代入方程则

```
>> syms x; t=3.5203; vpa(exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5)
ans =
    -.43167073997540938989914138801396e-4
```



(a) 某个区域的隐函数曲线　　(b) 局部放大后的求解结果

图 6-1　一元方程的图解法

可见，得出的 x 值处的函数值为 $-.4316\times10^{-4}$，故可见得出的根是原方程的根，但精度不是很高。用类似的方法还可以得出并验证其他的解。

6.1.1.2　二元方程的图解法

二元方程也是可以通过图解法求解的，可以通过 ezplot() 函数将第一个方程对应的曲线绘制出来，再使用 hold on 命令保持图形不被刷新，最后调用 ezplot() 函数将第二条曲线在同一坐标系下绘制出来。得出曲线后就可以通过读取交点坐标的方式得出联立方程的根。

【例 6-2】用图解法求解下面的联立方程。

$$\begin{cases} x^2\mathrm{e}^{-xy^2/2}+\mathrm{e}^{-x/2}\sin(xy)=0 \\ x^2\cos(x+y^2)+y^2\mathrm{e}^{x+y}=0 \end{cases}$$

【求解】利用隐函数图形绘制的方法，可以用图解法直接求解二元方程组，则可以通过下面的语句绘制出第一个方程的曲线，如图 6-2 (a) 所示。

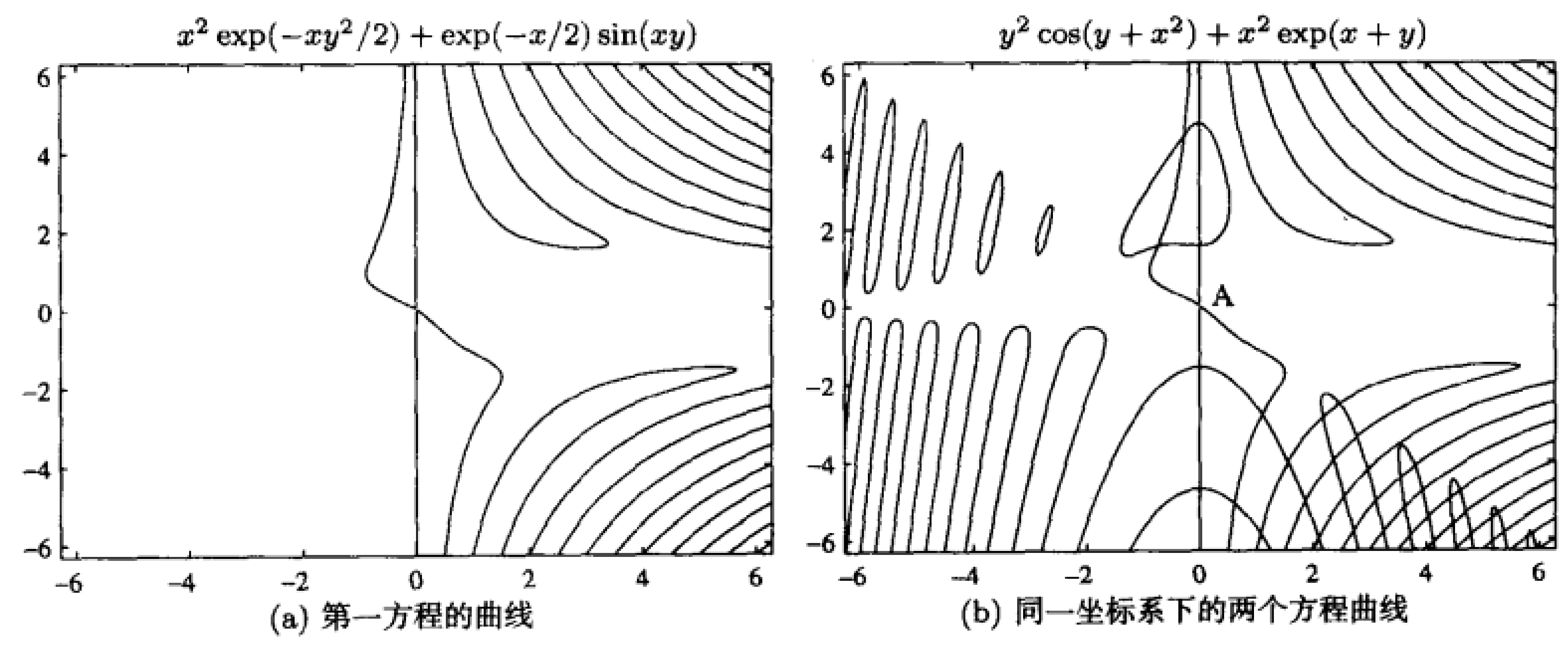
```
>> ezplot('x^2*exp(-x*y^2/2)+exp(-x/2)*sin(x*y)') % 第一个方程曲线
```

该曲线上所有的点均满足第一个方程。可以用 hold on 语句保护当前的坐标系，再用 ezplot() 函数绘制第二个方程的曲线，这样在同一坐标系下绘制出两条曲线，如图 6-2 (b) 所示。

```
>> hold on % 保护当前坐标系
   ezplot('y^2 *cos(y+x^2) +x^2*exp(x+y)')
```

这两个方程对应曲线的交点就是联立方程的解，所以可以通过图解法来求取二元联立方程的全部实根。但应该指出，这样求出的交点可能有的不是方程的根。观察图 6-2 (a) 可见，满足第一

方程的曲线自身就有交点，该交点不一定能满足第二方程，所以图 6-2 (b) 中的 A 点不是方程的根。



(a) 第一方程的曲线　　(b) 同一坐标系下的两个方程曲线

图 6-2　联立方程图解法示意图

6.1.2　多项式型方程的准解析解法

在介绍多项式方程的一般解法之前，先考虑下面给出的一个例子。

【例 6-3】试用图解方法求解下面的二元方程。

$$\begin{cases} x^2+y^2-1=0 \\ 0.75x^3-y+0.9=0 \end{cases}$$

【求解】用图解方法，可以用下面的语句直接绘制出两条曲线，如图 6-3 所示。曲线的交点就是原联立方程的解。

```
>> ezplot('x^2+y^2-1'); hold on   % 绘制第一方程的曲线
   ezplot('0.75*x^3-y+0.9')       % 绘制第二方程
```

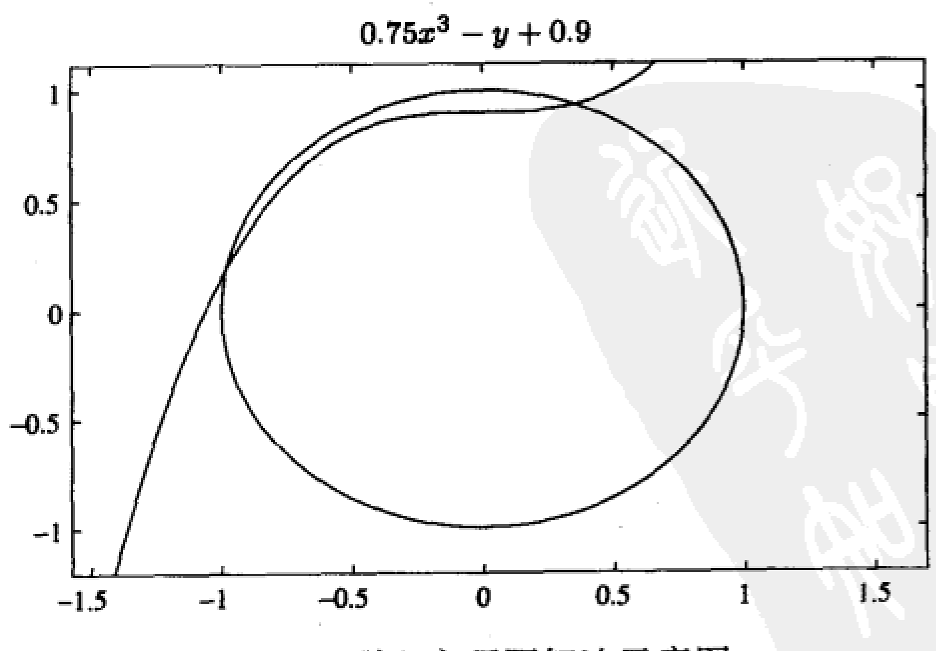


图 6-3　联立方程图解法示意图

从图 6-3 可见，这两条曲线共有两个交点，故可能轻易得出结论，原联立方程有两对根。事实上，这样的结论是错误的，由第 2 个方程可以显式地将 y 写成 x^3 的形式，代入第 1 方程则得出关于 x 的 6 次多项式方程，该方程应该有 6 对根。因为用二维图形只能求解出方程的实根，而不能求解出方程的复数根，所以利用图解法有时也能得出错误的结论。

一般多项式方程的根可以为实数，也可以为复数。MATLAB 符号运算工具箱中给出的 solve() 函数对多项式类方程是十分有效的，可以用该函数求解出多项式方程所有的根。该函数的定义格式为

S=solve(eqn$_1$,eqn$_2$,$\cdots$,eqn$_n$)	最简调用方式
[x,$\cdots$]=solve(eqn$_1$,eqn$_2$,$\cdots$,eqn$_n$)	直接得出根
[x,$\cdots$]=solve(eqn$_1$,eqn$_2$,$\cdots$,eqn$_n$,'x',$\cdots$')	同上，并指定变量

其中，eqn$_i$ 为第 i 个方程的符号表达式，可以同时求解若干个联立方程，还可以按照第 3 种调用格式显式地指出需要求解方程的变量名，第 2、3 种调用格式将求出方程的根，并按各个变量名返回到 MATLAB 的工作空间，第 1 种调用格式将返回一个结构体型变量 S，其各个成员变量，如 S.x 表示方程的根。

【例 6-4】 试用 solve() 函数求取例 6-3 中给出的联立方程。

【求解】 考虑上述的方法，利用 solve() 函数在 MATLAB 中可以给出如下的命令：

```
>> syms x y;
   [x,y]=solve('x^2+y^2-1=0','75*x^3/100-y+9/10=0')
x =
  [                                      .35696997189122287798839037801365]
  [  .86631809883611811016789809418650+1.2153712664671427801318378544391*i]
  [ -.55395176056834560077984413882735+.35471976465080793456863789934944*i]
  [                                     -.98170264842676789676449828873194]
  [ -.55395176056834560077984413882735-.35471976465080793456863789934944*i]
  [  .86631809883611811016789809418650-1.2153712664671427801318378544391*i]
y =
  [                                      .93411585960628007548796029415446]
  [ -1.4916064075658223174787216959259+.70588200721402267753918827138837*i]
  [  .92933830226674362852985276677202+.21143822185895923615623381762210*i]
  [                                      .19042035099187730240977756415289]
  [  .92933830226674362852985276677202-.21143822185895923615623381762210*i]
  [ -1.4916064075658223174787216959259-.70588200721402267753918827138837*i]
```

显然，这样得出的方程阶次太高，不存在解析解。然而，利用 MATLAB 的符号运算工具箱可以得出原始问题的高精度数值解，故这里称之为准解析解。可以看出，除了前面得出的两组实数根外，还得出了另外 4 组复数根，这是用普通数值解法所得不出来的。下面验证一下这样得出的根是不是原方程的根。直接给出下面的语句则可以发现，得出的根基本满足原方程。

```
>> [eval('x.^2+y.^2-1') eval('75*x.^3/100-y+9/10')]
```

```
ans =
    [          -.1e-31,              0.]
    [ .5e-30-.1e-30*i,           0.+0.*i]
    [         0.+0.*i,           0.+0.*i]
    [              0.,               0.]
    [         0.+0.*i,           0.+0.*i]
    [ .5e-30+.1e-30*i,           0.+0.*i]
```

【例 6-5】 多元多项式方程也可以用 solve() 函数直接求解。试求解下面的联立方程。

$$\begin{cases} x+3y^3+2z^2=1/2 \\ x^2+3y+z^3=2 \\ x^3+2z+2y^2=2/4 \end{cases}$$

【求解】 给出的联立方程是关于 x,y,z 的三元联立方程，从方程本身可见它只含有多项式项，所以从理论上说可以将之转换成一元的多项式方程，故方程可以由下面的 MATLAB 语句求出高精度数值解。

```
>> [x,y,z]=solve('x+3*y^3+2*z^2=1/2','x^2+3*y+z^3=2','x^3+2*z+2*y^2=2/4')
```

事实上，该方程最终被变换成 27 次的多项式方程，所以得出的解向量 $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}$ 均为有 27 个分量的向量。由于篇幅所限，在这里不列出全部的解，只列出其中一个根。

$$\begin{cases} x_1=-1.0869654762986136074917644096117 \\ y_1=0.03726466845064437552775081129721 \\ z_1=0.89073290972562790151300874796949 \end{cases}$$

可以用下面的语句验证解的误差是很小的，故而得出的结果是很精确的。

```
>> err=[x+3*y.^3+2*z.^2-1/2, x.^2+3*y+z.^3-2, x.^3+2*z+2*y.^2-2/4];
   norm(double(eval(err)))
ans =
    6.914598774554204e-026
```

其实，方程中若干式子也可以写成多项式乘积的形式。例如，联立方程最后一个式子改写成 $x^3+2zy^2=2/4$，其中含有非线性项 zy^2，这样的方程也能通过 solve() 函数直接求解，这时只需将求解语句变为

```
>> [x,y,z]=solve('x+3*y^3+2*z^2=1/2','x^2+3*y+z^3=2','x^3+2*z*y^2=2/4');
   err=[x+3*y.^3+2*z.^2-1/2, x.^2+3*y+z.^3-2, x.^3+2*z.*y.^2-2/4];
   norm(double(eval(err)))   % 将解代入求误差
ans =
    4.307698888228235e-026
```

就可以得出精确的解。经上述检验也得出了解为正确的结论。

【例 6-6】试求解下面的方程，其中含有自变量的倒数等形式。

$$\begin{cases} \dfrac{1}{2}x^2+x+\dfrac{3}{2}+2\dfrac{1}{y}+\dfrac{5}{2y^2}+3\dfrac{1}{x^3}=0 \\ \dfrac{y}{2}+\dfrac{3}{2x}+\dfrac{1}{x^4}+5y^4=0 \end{cases}$$

【求解】这样的方程指望求取解析解是基本上不可能的，但用下面的语句可以直接得出原方程的精确数值解，共有 26 对根。

```
>> syms x y;
   [x,y]=solve('x^2/2+x+3/2+2/y+5/(2*y^2)+3/x^3=0',...
          'y/2+3/(2*x)+1/x^4+5*y^4','x,y');
   size(x)
ans =
     26      1
```

其中，第 1 对根为

$$\begin{cases} x_1=0.93020851860976459141084889836970+\mathrm{j}0.44242013491916995256639798767139 \\ y_1=-0.39388334385234983573088749775055+\mathrm{j}0.62771855170677843775952969855352 \end{cases}$$

将得出的全部方程根其代入原始方程，则能得出很小的计算误差，达到 10^{-30} 级，说明该方程的各个解的正确性。

```
>> err=[x.^2/2+x+3/2+2./y+5./(2*y.^2)+3./x.^3,y/2+3./(2*x)+1./x.^4+5*y.^4];
   norm(double(eval(err)))
ans =
    8.247060852664867e-030
```

【例 6-7】试求解下面带有参数的方程。

$$\begin{cases} x^2+ax^2+6b+3y^2=0 \\ y=a+x+3 \end{cases}$$

【求解】MATLAB 符号运算工具箱中提供的 solve() 函数还可以直接实现带有变量的方程的解，这样的求解用普通的数值解方法是不能实现的。求解上述方程只需给出下面语句即可。

```
>> syms a b x y; [x,y]=solve('x^2+a*x^2+6*b+3*y^2=0','y=a+(x+3)','x,y')
x =
   [ 1/2/(4+a)*(-6*a-18+2*(-21*a^2-45*a-27-24*b-6*a*b-3*a^3)^(1/2))]
   [ 1/2/(4+a)*(-6*a-18-2*(-21*a^2-45*a-27-24*b-6*a*b-3*a^3)^(1/2))]
y =
   [ a+1/2/(4+a)*(-6*a-18+2*(-21*a^2-45*a-27-24*b-6*a*b-3*a^3)^(1/2))+3]
   [ a+1/2/(4+a)*(-6*a-18-2*(-21*a^2-45*a-27-24*b-6*a*b-3*a^3)^(1/2))+3]
```

用 LaTeX 可以表示解的数学式子为

$$x=\frac{-6a-18\pm 2\sqrt{-21a^2-45a-27-24b-6ab-3a^3}}{2(4+a)},\quad y=a+x+3$$

其实，该方法同样适用于更高阶方程的解，但得出的解是很冗长的，不适合显示出来。

然而，解析求解的方法并不是万能的，因为这里的例子最终可以转换为一元多项式方程，所以能用它求解，但更一般的方程是不能解出的。

6.1.3 一般非线性方程数值解

MATLAB 语言环境中提供了 fsolve() 函数，能够求出已知多元方程的一个实数根。该函数的调用格式为

```
x=fsolve(Fun,x0)                                   最简求解语句
[x,f,flag,out]=fsolve(Fun,x0,opt,p1,p2,...)        一般求解格式
```

其中，Fun 为描述需求解的方程 M-函数，$\boldsymbol{x}_0$ 为搜索点的初值，方程求根程序将从该值开始以逐步减小误差的算法搜索出满足方程的实根 $\boldsymbol{x}$。若返回的 flag 大于 0，则表示求解成功，否则求解出现问题。

对于更复杂的问题，用户可以定义方程求解控制参数 opt 来控制求解方法或其他要求，更好地得出方程的根。该变量是一个结构体数据，其常用的成员变量在表 6-1 中给出，用户可以用下面的语句修改控制变量。

表 6-1 方程求解与最优化的控制参数表

参数名	参数说明
Display	中间结果显示方式，其值可以取 off 表示不显示中间值，iter 表示逐步显示，notify 表示在求解不收敛时给出提示，final 只显示最终值
GradObj	求解最优化问题时使用，表示目标函数的梯度是否已知，可以选择为 'off' 或 'on'
LargeScale	表示是否使用大规模问题算法，取值为 on 或 off，一般几个变量的问题不必采用该算法
MaxIter	方程求解和优化过程最大允许的迭代次数，若方程未求出解，可以适当增加该值
MaxFunEvals	方程函数或目标函数的最大调用次数
TolFun	误差函数误差限控制量，当函数的绝对值小于此值即终止求解
TolX	解的误差限控制量，当解的绝对值小于此值即终止求解

```
opt=optimset;                                 获得默认的常用变量
opt.TolX=1e-10; 或 set(opt,'TolX',1e-10)      用这两种方法修改参数
```

其中的某些值，如 MaxFunEvals 和问题类型有关，如方程求解和有约束最优化问题一般选择其值为 100 倍的自变量个数，而无约束最优化问题一般支持 200 倍的变量个数。

【例 6-8】 Lambert 函数是一类特殊函数，该方程的数学表示为 $w = \mathrm{lam}(x)$，其中，x 为自变量，w 为 Lambert 方程的解 $we^w = x$，对不同的 x 值，试求解 Lambert 方程得出 w 值，并绘制出 x 和 w 之间的关系曲线。

【求解】 通过循环的方法对不同的 x 值进行求解，就可以绘制出 Lambert 函数的曲线，如图 6-4 所示。

```
>> y=[]; xx=0:.05:10; x0=0; h=optimset; h.Display='off';
   for x=xx
      f=inline(['w.*exp(w)-' num2str(x)],'w');
      y1=fsolve(f,x0,h); x0=y1; y=[y,y1];
   end
   plot(xx,y)
```

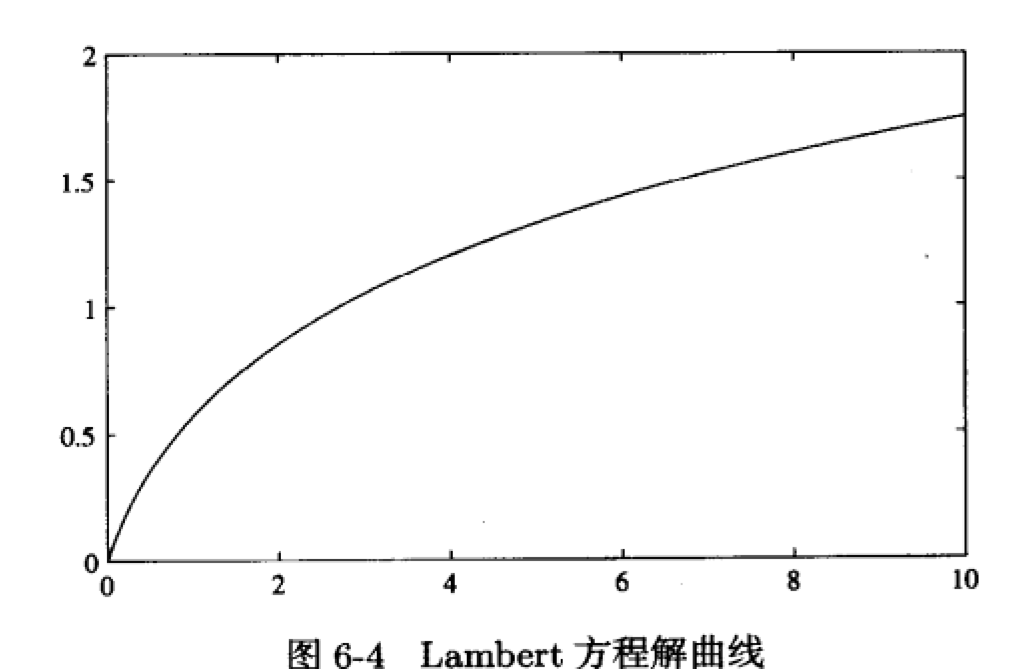


图 6-4 Lambert 方程解曲线

MATLAB 的符号运算工具箱中提供了一个 Lambert 函数求值函数 lambertw()，可以用来直接求取 Lambert 函数的值。用下面语句可以直接绘制出与图 6-4 完全一致的曲线。

```
>> y0=lambertw(xx); plot(xx,y0)
```

【例 6-9】 试用数值方法求解例 6-3 中给出的二元方程。

【求解】 令 $p_1 = x, p_2 = y$，可以编写出一个描述此二元方程的函数如下：

```
function q = my2deq(p)
q=[p(1)*p(1)+p(2)*p(2)-1; 0.75*p(1)^3-p(2)+0.9];
```

这样，就可以在给定的初值 $x_0 = 1, y_0 = 2$ 下调用 fsolve() 函数，直接求出方程的根。

```
>> OPT=optimset; OPT.LargeScale='off';
   [x,Y,c,d] = fsolve('my2deq',[1; 2],OPT),
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.
x =                              Y = 1.0e-009 *
   0.35696997213260                 0.12151502026825
   0.93411585957908                 0.09640666043254
c =
     1
d =
    iterations: 6
     funcCount: 21
```

```
        algorithm: 'trust-region dogleg'
    firstorderopt: 1.306115559429460e-010
```

可以看出，在求解此二元方程时仅调用了方程函数 21 次就得出了方程的解，把解回代到原始方程中也得到了较高精度的结果 (1.3×10^{-10})。

同样，可以建立如下的 inline() 函数，由下面的语句也可以得出相同的结果。

```
>> f=inline('[p(1)*p(1)+p(2)*p(2)-1; 0.75*p(1)^3-p(2)+0.9]','p');
   [x,Y] = fsolve(f,[1; 2],OPT)   % 结果和上述完全一致，从略。
```

若采用 MATLAB 7.0 开始引入的匿名函数，描述方程可以更简洁

```
>> f=@(p)[p(1)*p(1)+p(2)*p(2)-1; 0.75*p(1)^3-p(2)+0.9];
```

可见，引入 inline() 函数或匿名函数，无需为要求解的每个数学问题都编写一个单独的 MATLAB 模型文件，这样使得问题的求解与文件管理变得更容易、方便。

若改变初始猜测值，令 $\boldsymbol{x}_0=[-1,0]^{\mathrm{T}}$，则

```
>> [x,Y,c,d]=fsolve(f,[-1,0]',OPT); x, Y, kk=d.funcCount
x =                          Y =  1.0e-010 *
   -0.98170264846110              0.56187055008650
    0.19042035096244             -0.44996673054243
kk =
    15
```

可见，初值改变之后，还能得出另外一组解。所以初值的选择有时对整个问题的求解有很大的影响，在某些初值下甚至无法搜索到方程的解。

【例 6-10】 重新考虑例 6-1 中给出的方程 $\mathrm{e}^{-3t}\sin(4t+2)+4\mathrm{e}^{-0.5t}\cos(2t)=0.5$，试用数值方法求其更精确的数值解。

【求解】 显然该非线性方程没有解析解。例 6-1 中用图解法得出了一个数值解为 $t=3.5203$，以此解为初值，则可以用 fsolve() 函数得出更精确的数值解。

先使用符号运算工具箱中的 solve() 函数求解此问题。

```
>> syms t x; solve(exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5)
ans =
   .67374570500134756702960220427474
```

可见，这样的方法不允许手工选择初始搜索点，所以只能获得这样的结果。用图解法选择初值，再用数值解法可以得出任意初值下的数值解。

```
>> y=inline('exp(-3*t).*sin(4*t+2)+4*exp(-0.5*t).*cos(2*t)-0.5','t');
   ff=optimset; ff.Display='iter'; [t,f]=fsolve(y,3.5203,ff)
                                          Norm of      First-order   Trust-region
 Iteration  Func-count     f(x)           step         optimality    radius
     0          2      1.8634e-009                     5.16e-005           1
     1          4     3.67694e-019  3.61071e-005       7.25e-010           1
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.
```

```
  t =                          f =
     3.52026389294877             -6.063776702980306e-010
```

代入原方程发现，该解的精度明显高于图解法的精度，且可以任意选择初值。从算法本身看，默认的求解精度偏低，所以可以试着重新设置相关的控制变量，就可以得出下面更精确的结果。

```
>> ff=optimset; ff.TolX=1e-16; ff.TolFun=1e-30;
   ff.Display='iter'; [t,f]=fsolve(y,3.5203,ff)
                                        Norm of      First-order   Trust-region
 Iteration  Func-count     f(x)           step         optimality    radius
     0          2      1.8634e-009                     5.16e-005          1
     1          4     3.67694e-019   3.61071e-005      7.25e-010          1
     2          6                0   5.07218e-010              0          1
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.
t =                          f =
    3.52026389244155             0
```

从求解效果看，给出一个很精确的初值对整个求解速度和精度没有太大的帮助，真正使得本例子获得高精度解的还是 TolFun 属性。

6.2 无约束最优化问题求解

无约束最优化问题是最简单的一类最优化问题，其一般数学描述为

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{6-2-1}$$

其中，$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$ 称为优化变量，$f(\cdot)$ 函数称为目标函数，该数学表示的含义亦即求取一组 $\boldsymbol{x}$ 向量，使得最优化目标函数 $f(\boldsymbol{x})$ 为最小，故这样的问题又称为最小化问题。其实，最小化是最优化问题的通用描述，它不失普遍性。如果要想求解最大化问题，那么只需给目标函数 $f(\boldsymbol{x})$ 乘一个负号就能立即将最大化问题转换成最小化问题。所以本书中描述的全部问题都是最小化问题。

6.2.1 解析解法和图解法

无约束最优化问题的最优点 $\boldsymbol{x}^*$ 处，目标函数 $f(\boldsymbol{x})$ 对 $\boldsymbol{x}$ 各个分量的一阶导数为 0，从而可以列出下面的方程：

$$\left.\frac{\partial f}{\partial x_1}\right|_{\boldsymbol{x}=\boldsymbol{x}^*}=0,\ \left.\frac{\partial f}{\partial x_2}\right|_{\boldsymbol{x}=\boldsymbol{x}^*}=0,\ \cdots,\ \left.\frac{\partial f}{\partial x_n}\right|_{\boldsymbol{x}=\boldsymbol{x}^*}=0 \tag{6-2-2}$$

求解这些方程构成的联立方程可以得出极值点。其实，解出的一阶导数均为 0 的极值点不一定都是极小值的点，其中有的还可能是极大值点。极小值问题还应该有正的二阶导数。对于单变量的最优化问题，可以考虑采用解析解的方法进行求解。然而多变量最优

化问题因为需要将其转换成求解多元非线性方程，其难度也不低于直接求取最优化问题，所以没有必要采用解析解方法求解。

一元函数最优化问题的图解法也是很直观的，应绘制出该函数的曲线，在曲线上就能看出其最优值点。二元函数的最优化也可以通过图解法求出。但三元或多元函数，由于用图形没有办法表示，所以不适合用图解法求解。

【例 6-11】 例 6-1 中给出的方程 $f(t)=\mathrm{e}^{-3t}\sin(4t+2)+4\mathrm{e}^{-0.5t}\cos(2t)-0.5$，试用解析求解和图形求解的方法研究该函数的最优性。

【求解】 可以先表示该函数，并解析地求解该函数的一阶导数，用 ezplot() 函数可以绘制出 $t\in[0,4]$ 区间内一阶导函数的曲线，如图 6-5 (a) 所示。

```
>> syms t; y=exp(-3*t)*sin(4*t+2)+4*exp(-0.5*t)*cos(2*t)-0.5;
   y1=diff(y,t); % 求取一阶导函数
   ezplot(y1,[0,4])  % 绘制出选定区间内一阶导函数曲线
```

其实，求解导函数等于 0 的方程不比直接求解其最优值简单。用图解法可以看出，在这个区间内有两个点，A_1 和 A_2，使得它们的一阶导函数为 0，但从其一阶导数走向看，A_2 点对应负的二阶导数值，所以该点对应于极大值点，而 A_1 点对应于正的二阶导数值，故为极小值点。A_1 点的值可以由下面的语句直接解出。

```
>> t0=solve(y1), ezplot(y,[0,4])     % 求出一阶导数等于零的点
t0 =
   1.4528424981725411893375778048840
>> y2=diff(y1); b=subs(y2,t,t0) % 并验证二阶导数为正
b =
   7.8553420253333601379464405534590
```

这样，就可以求出函数的最小值。还可以用图形绘制的方法进一步验证得出的结果，如图 6-5 (b) 所示，可见，A_1 为最小值，A_2 为最大值。

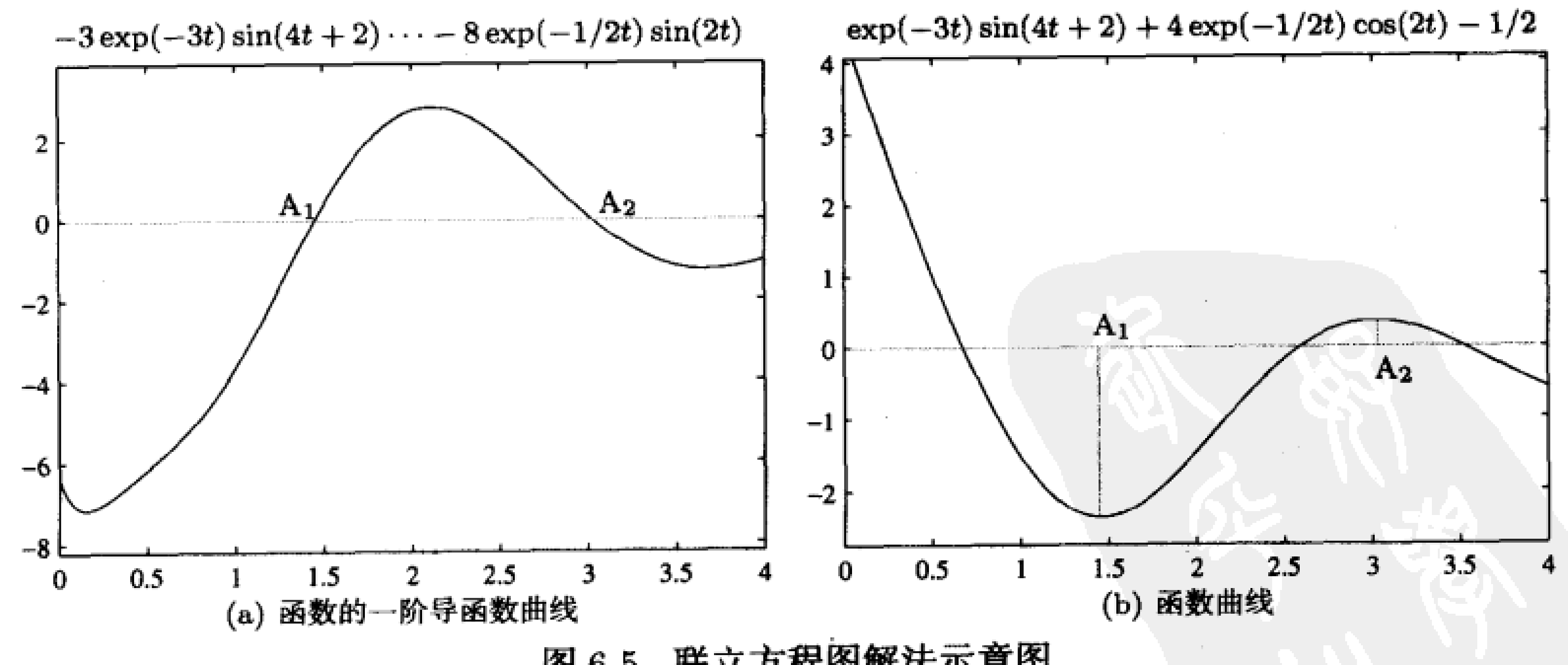


图 6-5 联立方程图解法示意图

然而因为给定的函数是非线性函数，所以用解析法或类似的方法求解最小值问题一点都不比直接求解最优化问题简单。因此，除演示之外，不建议用这样的方法求解该问题，而直接采用最

优化问题求解程序得出问题的解。

6.2.2 基于 MATLAB 的数值解法

MATLAB 语言中提供了求解无约束最优化的函数 fminsearch()，其最优化工具箱中还提供了函数 fminunc()，二者的调用格式完全一致，为

x=fminunc(Fun,x_0)	最简求解语句
[x,f,flag,out]=fminunc(Fun,x_0,opt,p_1,p_2,$\cdots$)	一般求解格式

其输入与返回参数的定义与 fsolve() 函数中的控制变量完全一致。该函数主要采用了文献 [33] 中提出的单纯形算法。下面将通过例子来演示无约束最优化问题的数值解法。

【例 6-12】已知二元函数 $z = f(x,y) = (x^2-2x)\mathrm{e}^{-x^2-y^2-xy}$，试用 MATLAB 提供的求解函数求出其最小值，并用图形方法表示其求解过程。

【求解】因为函数中给出的自变量是 x, y，而最优化函数需要求取的是自变量向量 x，故在求解前应该先进行变量替换，如令 $x_1 = x, x_2 = y$，这样就可以用下面的语句由 inline() 形式定义出目标函数 f，然后将求解控制变量中的 Display 属性设置为 'iter'，这样可以显示中间的搜索结果。用下面的语句求解出最优解。

```
>> f=inline('(x(1)^2-2*x(1))*exp(-x(1)^2-x(2)^2-x(1)*x(2))','x');
   x0=[0; 0]; ff=optimset; ff.Display='iter';
   x=fminsearch(f,x0,ff)
 Iteration   Func-count     min f(x)          Procedure
     1           3        -0.000499937        initial
     2           4        -0.000499937        reflect
    ..          ...     ...............       .........
    72         137         -0.641424          contract outside
Optimization terminated successfully:
x =
    0.61105369700841
   -0.30557806047584
```

同样的问题用 fminunc() 函数求解，则可以得出如下的结果。

```
>> x=fminunc(f,[0;.0],ff)
                                                        Gradient's
 Iteration  Func-count        f(x)        Step-size    infinity-norm
     0           3              0                            2
     1           6         -0.367879         0.5           0.736
     2           9         -0.571873          1            0.483
     3          15         -0.632398      0.284069         0.144
     4          18         -0.638773          1            0.063
     5          21          -0.64141          1           0.00952
```

```
     6          24        -0.641424              1        0.000619
     7          27        -0.641424              1        1.79e-006
x =
   0.61104620481440
  -0.30552415363569
```

比较两种方法，显然可以看出，用 fminunc() 函数的效率明显高于 fminsearch()。所以在无约束最优化问题求解时，如果安装了最优化工具箱则建议使用 fminunc() 函数。

在求解过程中，如果手工修改 fminunc() 下级的 fminusub() 函数文件，就可以追踪出各个搜索中间点的坐标。下面在图形上显示出搜索中间过程。假设选择 $\boldsymbol{x}=[2,1]^{\mathrm{T}}$，则由下面的语句可以得出所需的解，同时还能由修改的函数得出中间点坐标为

$$
\begin{array}{ccccccccc}
x_i & 2 & 0.2401 & -0.1398 & 0.2168 & 0.3355 & 0.5514 & 0.6129 & 0.6111 \\
y_i & 1 & 1.0502 & 0.5752 & 1.0210 & -0.5508 & -0.1775 & -0.3053 & -0.3058
\end{array}
$$

用下面的语句可以绘制出搜索过程中间点的轨线，如图 6-6 所示。

```
>> xx=[2  0.2401 -0.1398 0.2168  0.3355  0.5514  0.6129  0.6111
       1  1.0502  0.5752 1.0210 -0.5508 -0.1775 -0.3053 -0.3058];
   [x,y]=meshgrid(-3:.1:3, -2:.1:2);
   z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
   contour(x,y,z,30); line(xx(1,:),xx(2,:))
   h=line(xx(1,:),xx(2,:)); set(h,'Marker','o')
```

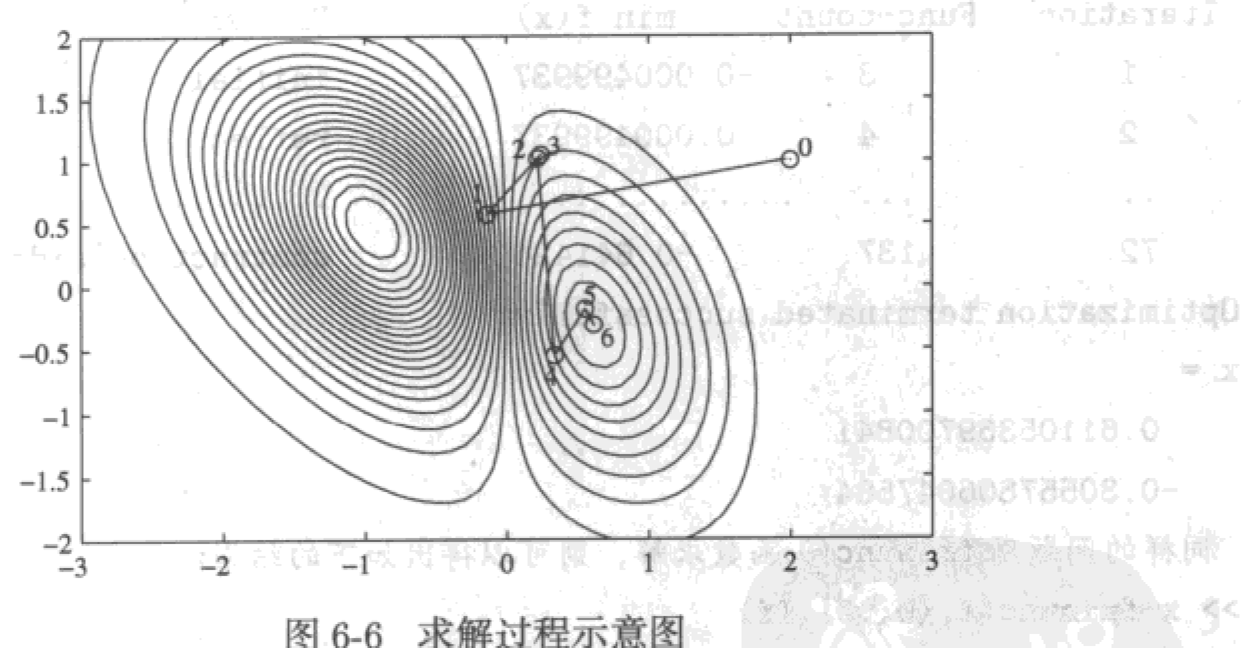


图 6-6 求解过程示意图

6.2.3 全局最优解与局部最优解

以单变量 x 为例，无约束最优化问题函数有解的必要条件是 $\mathrm{d}f(x)/\mathrm{d}x=0$，但满足该条件的 x 值可能不惟一，可能存在多个解。从最优化搜索的角度来说，可能找到其中一个这样的点。下面将通过例子引入全局最优解和局部最优解的概念。

【例 6-13】假设目标函数为 $y(t)=\mathrm{e}^{-2t}\cos 10t+\mathrm{e}^{-3t-6}\sin 2t$，$t\geqslant 0$，试观察不同的初值能得出的最小值，并讨论局部最小值与全局最小值的概念。

【求解】 由给定的目标函数，可以立即写出可以用于无约束最优化搜索的 MATLAB 表示，可以采用下面的 inline() 函数或编写相应的 MATLAB 文件。

```
>> f=inline('exp(-2*t).*cos(10*t)+exp(-3*(t+2)).*sin(2*t)','t'); % 目标函数
```

若选定初始搜索点为 $t_0=1$，则可以给出如下的语句获得目标函数的最优值为

```
>> t0=1; [t1,f1]=fminsearch(f,t0); [t1 f1]
ans =
   0.92275390625000         -0.15473299821860
```

若选择另一个初始点，则可以得出如下的最优解。

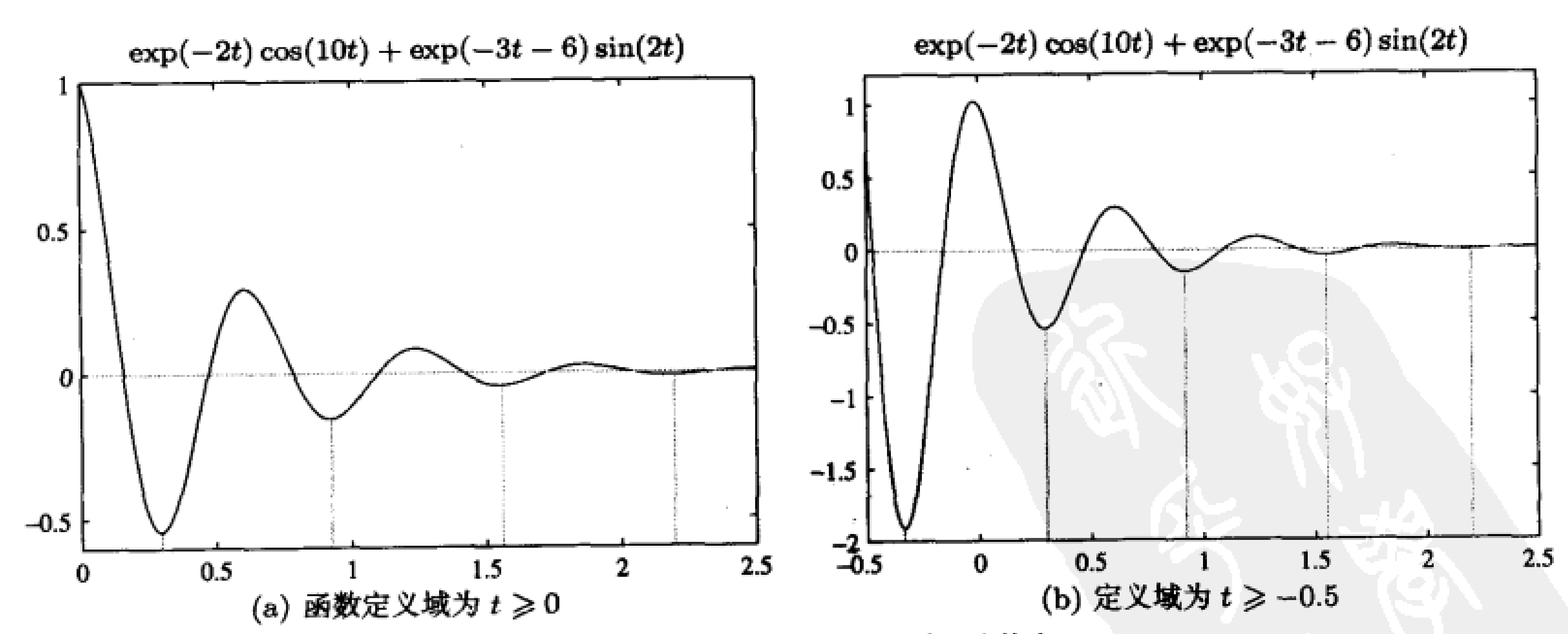
```
>> t0=0.1; [t2,f2]=fminsearch(f,t0); [t2 f2]
ans =
   0.29445312500000         -0.54362463738706
```

可见，给出不同的初值，此函数能得出不同的"最优解"，但从最优解处的函数值看，t_1 处的函数值显然大于 t_2 处的函数值。故可以得出结论，t_1 得出的并非真正的最优解，而是某种局部最优解。试凑其他的初值，如 $t_0=1.5,2,2.5,\cdots$ 还可能得出其他的局部最优值，这里不一一列出。用 MATLAB 的 ezplot() 函数可以绘制给定目标函数 $y(t)$ 在 $t\in(0,2.5)$ 定义域内的曲线，如图 6-7 (a) 所示，区间内的全局和局部最优值均由虚线表示出来。

```
>> syms t; y=exp(-2*t)*cos(10*t)+exp(-3*(t+2))*sin(2*t);
   ezplot(y,[0,2.5]); set(gca,'Ylim',[-0.6,1])
```

从图 6-7 (a) 可以看出，在 $t\geqslant 0$ 定义域内，t_2 点是目标函数真正的最优值，在最优化理论中又称为全局最优解，由于初值选择不同的值可能得出不同的最优值，其中有些是局部最优值，所以这里给出的 fminsearch() 函数并不能保证求出全局最优值。事实上，目前所有的最优化算法没有哪一种能保证能求出最优化问题的全局最优解。

现在再考虑更大些的定义域，即 $t\geqslant -0.5$，则用下面的语句能绘制出该函数在新定义域内的曲线，如图 6-7 (b) 所示。



(a) 函数定义域为 $t\geqslant 0$　　(b) 定义域为 $t\geqslant -0.5$

图 6-7 全局最优解与局部最优解

```
>> ezplot(y,[-0.5,2.5]); set(gca,'Ylim',[-2,1.2])
   t0=-0.2; [t3,f3]=fminsearch(f,t0); [t3 f3]
```

```
ans =
   -0.33398437500000       -1.91625251248676
```

从图 6-7 (b) 可见，前面得出的全局最优解在新的定义域内就不再是全局最优解了，比起新的最优解来说，它只是局部最优解，若进一步扩大定义域，则得出的 t_3 将也不再是全局最优解了，而称为局部最优解。若将定义域扩展到 $t \in (-\infty, \infty)$ 区间，则原函数将没有真正意义的全局最优解。

通过上面的例子可以看出，利用 MATLAB 的求解函数或其他现成的最优化问题求解函数可能得出局部最优解，而不是全局最优解，这就需要读者自己去试不同的初值，看看得出的最优解是否相同，如果不同，则比较一下哪个是局部最优解。遗传算法提供了一种同时试测不同初值的算法，在求解全局最优解上有一定的改进，但也不能保证得出全局最优解。有关遗传算法及其在最优化问题中的应用请具体参见第 10.3 节。

6.2.4 利用梯度求解最优化问题

有时最优化问题求解速度较慢，有时甚至无法搜索到较精确的最优点，尤其是变量较多的最优化问题，所以需要引入目标函数梯度，以加快计算速度，改进搜索精度。然而，有时计算梯度也是需要时间的，也会影响整个运算速度，所以实际求解时应该考虑是不是值得引入梯度的概念。

在利用 MATLAB 最优化工具箱求解最优化问题时，也应该和目标函数在同一函数中描述梯度函数，亦即这时 MATLAB 的目标函数应该返回两个变量，第一个变量仍然表示目标函数，第二个变量可以返回/梯度函数。同时，还应该将求解控制变量的 `GradObj` 属性设置成 `'on'`，这样就可以利用梯度来求解最优化问题了。

【例 6-14】 试求解 Rosenbrock 函数 $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ 的无约束最优化问题。

【求解】 从目标函数可以看出，由于它为两个平方数的和，所以当 $x_2 = x_1 = 1$ 时，整个目标函数有最小值 0。用下面语句可以绘制出目标函数的三维等高线图，如图 6-8 所示。

```
>> [x,y]=meshgrid(0.5:0.01:1.5); z=100*(y.^2-x).^2+(1-x).^2;
   contour3(x,y,z,100), set(gca,'zlim',[0,310])
```

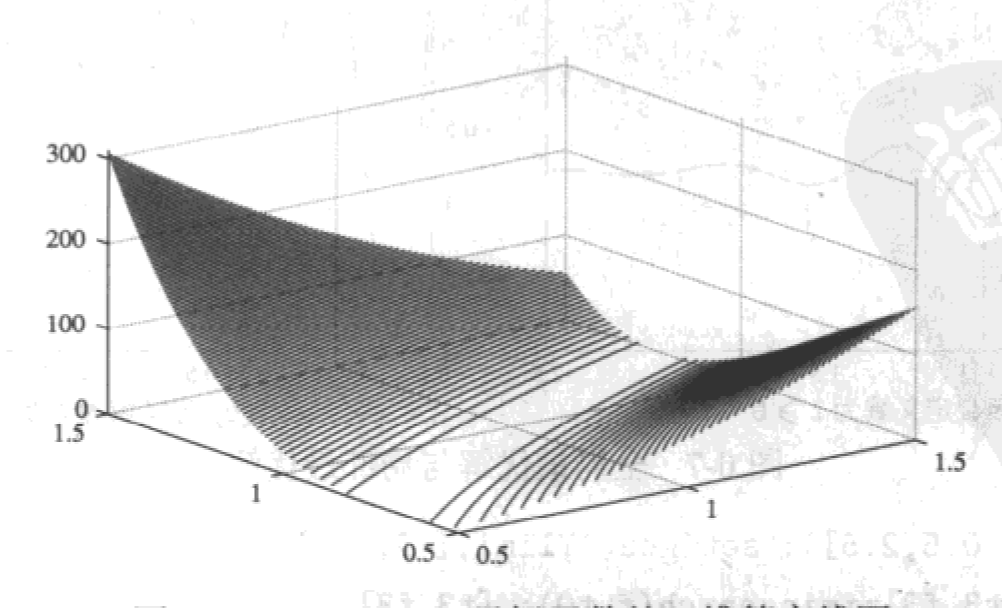


图 6-8　Rosenbrock 目标函数的三维等高线图

由得出的曲线看，其最小值点在图中的一个很窄的白色带状区域内，故 Rosenbrock 目标函数又称为香蕉形函数，而在这个区域内的函数值变化较平缓，这就给最优化求值带来很多麻烦。该函数经常用来测试最优化算法的优劣。现在观察用下面语句求解最优化问题。

```
>> f=inline('100*(x(2)-x(1)^2)^2+(1-x(1))^2','x');
   ff=optimset; ff.TolX=1e-10; ff.TolFun=1e-20; ff.Display='iter';
   x=fminunc(f,[0;0],ff)
```

Iteration	Func-count	f(x)	Step-size	Gradient's infinity-norm
0	3	1		2
1	12	0.771192	0.0817341	5.34
...	...	....	....	....
19	78	6.73881e-011	1	7.24e-005
20	81	1.94711e-011	1	1.06e-006

```
x =
   0.99999558847268
   0.99999116718532
```

可见，该算法无法在给定的步数内精确搜索到真值 (1,1)，用传统的最速下降法更无法搜索到真值。这时需要引入梯度的概念。

对给定的 Rosenbrock 函数，利用符号运算工具箱即可以求出其梯度向量为

```
>> syms x1 x2; f=100*(x2-x1^2)^2+(1-x1)^2;
   J=jacobian(f,[x1,x2])
J =
   [ -400*(x2-x1^2)*x1-2+2*x1,          200*x2-200*x1^2]
```

这时，可以在目标函数中描述其梯度，故需要重新编写目标函数为

```
function [y,Gy]=c6fun3(x)
y=100*(x(2)-x(1)^2)^2+(1-x(1))^2; % 目标函数
Gy=[-400*(x(2)-x(1)^2)*x(1)-2+2*x(1); 200*x(2)-200*x(1)^2]; % 梯度
```

这样，就应该给出如下命令来求解最优化问题。

```
>> ff.GradObj='on'; x=fminunc('c6fun3',[0;0],ff)
```

Iteration	f(x)	Norm of step	First-order optimality	CG-iterations
0	1		2	
1	1	1	2	1
2	0.953125	0.25	12.5	0
..	....	....	....	...
17	9.28362e-025	2.64127e-007	3.06e-012	1
18	2.03181e-028	2.11575e-012	2.84e-014	1

```
Optimization terminated successfully:
```

```
Relative function value changing by less than OPTIONS.TolFun
x =
   0.99999999999999
   0.99999999999997
```

可见，引入了梯度则可以明显加快搜索的进度，且最优解也基本上逼近于真值，这是不使用梯度不可能得到的，所以从本例可以看出梯度在搜索中的作用。然而，在有些例子中引入梯度也不是很必要，因为梯度本身的计算和编程需要更多的时间。

6.3　有约束最优化问题的计算机求解

有约束最优化问题的一般描述为

$$\min_{\boldsymbol{x}\ \text{s.t.}\ \boldsymbol{G}(\boldsymbol{x})\leqslant \boldsymbol{0}} f(\boldsymbol{x}) \tag{6-3-1}$$

其中，$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$，记号 s.t. 是英文 subject to 的缩写，表示满足后面的关系。该数学表示的含义为求取一组 $\boldsymbol{x}$ 向量，使得在满足约束条件 $\boldsymbol{G}(\boldsymbol{x})\leqslant 0$ 的前提下能够使目标函数 $f(\boldsymbol{x})$ 最小化。在实际遇到的最优化问题中，有时约束条件可能是很复杂的，它既可以是等式约束，也可以是不等式约束等，既可以是线性的，也可能是非线性的，有时甚至可以不能用纯数学函数来描述。

6.3.1　约束条件与可行解区域

满足约束条件 $\boldsymbol{G}(\boldsymbol{x})\leqslant \boldsymbol{0}$ 的 $\boldsymbol{x}$ 范围称为可行解区域 (feasible region)。下面通过例子演示二元问题的可行解范围与图解结果。

【例 6-15】考虑下面二元最优化问题的求解，试用图解方法对该问题进行研究。

$$\max_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}9\geqslant x_1^2+x_2^2\\ x_1+x_2\leqslant 1\end{array}\right.} (-x_1^2-x_2)$$

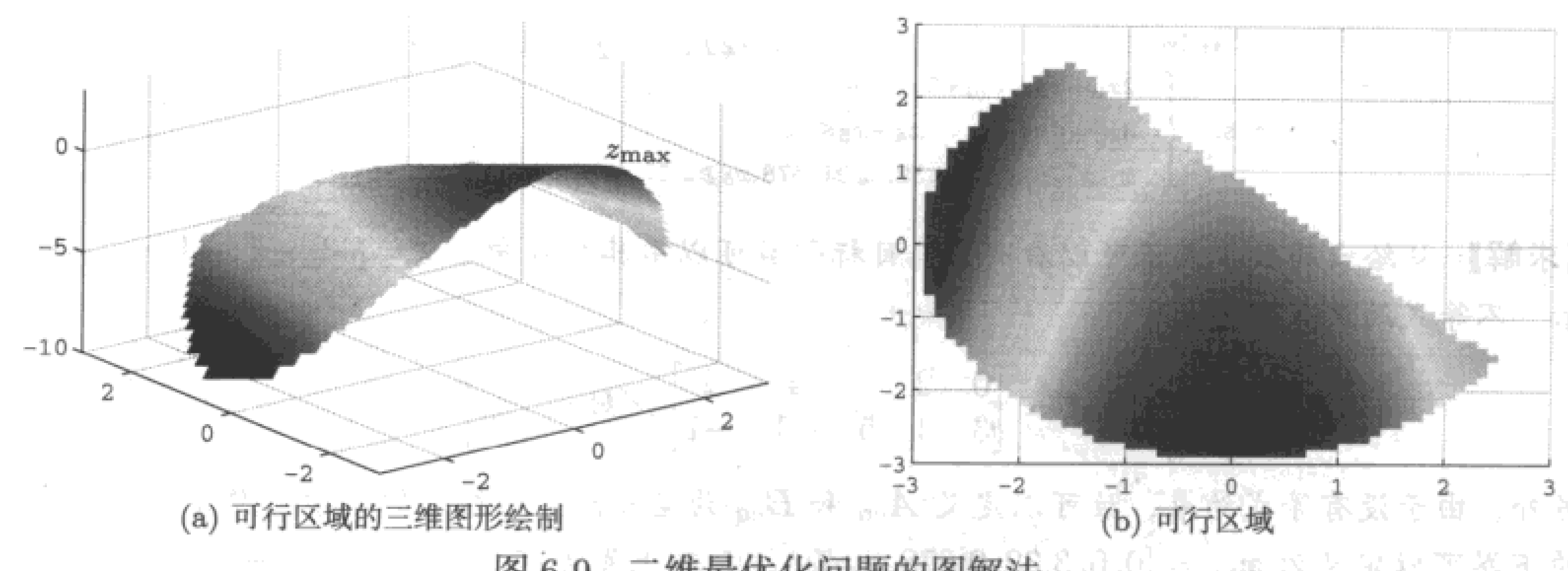
【求解】由约束条件可见，若在 $[-3,3]$ 区间选择网格，则可以得出无约束时目标函数的三维图形数据。可以用下面的语句获得这些数据。

```
>> [x1,x2]=meshgrid(-3:.1:3);  % 生成网格型矩阵
   z=-x1.^2-x2;                % 计算出矩阵上各点的高度
```

引入了约束条件，则在图形上需要将约束条件以外的点剔除掉。第 2.5 节中介绍了如何剔除三维图形中点的方法，具体的方法是找出这些点的横纵坐标值，将其函数值设置成不定式 NaN 即可剔除这些坐标点。这样可以使用如下的语句进行求解。

```
>> i=find(x1.^2+x2.^2>9); z(i)=NaN;  % 找出 x1^2+x2^2>9 的坐标，并置函数值为 NaN
   i=find(x1+x2>1); z(i)=NaN;  % 找出 x1+x2>1 的坐标，并置函数值为 NaN
   surf(x1,x2,z); shading interp;
```

该语句可以直接绘制出如图 6-9 (a) 所示的三维图形，若想从上向下观察该图形，则可以使用 `view(0,90)` 命令，这样可以得出如图 6-9 (b) 所示的二维投影图。图形上的区域为相应最优化问题的可行区域，即满足约束条件的区域。该区域内对应目标函数的最大值就是原问题的解，故从图形可以直接得出结论，问题的解为 $x_1=0, x_2=3$，用 `max(z(:))` 可以得出最大值为 3。



(a) 可行区域的三维图形绘制　　(b) 可行区域

图 6-9　二维最优化问题的图解法

对于一般的一元问题和二元问题，可以用图解法直接得出问题的最优解。但对于一般的多元问题和较复杂的问题，则不适合用图解法求解，而只能用数值解的方法进行求解，也没有检验全局最优性的方法。

6.3.2　线性规划问题的计算机求解

线性规划问题是一类特殊的问题，也是最简单的有约束最优化问题。在线性规划中，目标函数和约束函数都是线性的，其整个问题的数学描述为

$$\min_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}\boldsymbol{A}\boldsymbol{x}\leqslant\boldsymbol{B}\\ \boldsymbol{A}_{\rm eq}\boldsymbol{x}=\boldsymbol{B}_{\rm eq}\\ \boldsymbol{x}_m\leqslant\boldsymbol{x}\leqslant\boldsymbol{x}_M\end{array}\right.} \boldsymbol{f}^{\rm T}\boldsymbol{x} \tag{6-3-2}$$

为描述原问题的方便及求解的高效性起见，这里的约束条件已经进一步细化为线性等式约束 $\boldsymbol{A}_{\rm eq}\boldsymbol{x}=\boldsymbol{B}_{\rm eq}$，线性不等式约束 $\boldsymbol{A}\boldsymbol{x}\leqslant\boldsymbol{B}$，$\boldsymbol{x}$ 变量的上界向量 $\boldsymbol{x}_M$ 和下界向量 $\boldsymbol{x}_m$，使得 $\boldsymbol{x}_m\leqslant\boldsymbol{x}\leqslant\boldsymbol{x}_M$。

对不等式约束来说，MATLAB 定义的标准型是 “⩽” 关系式。如果约束条件中某个式子是 “⩾” 关系式，则在不等号两边同时乘以 -1 就可以转换成 “⩽” 关系式了。

线性规划是一类最简单的有约束最优化问题，求解线性规划问题有多种算法。其中，单纯形法是最有效的一种方法，MATLAB 的最优化工具箱中实现了该算法，提供了求解线性规划问题的 `linprog()` 函数。该函数的调用格式为

$[\boldsymbol{x},f_{\rm opt},\texttt{flag},\texttt{c}]=\texttt{linprog}(\boldsymbol{f},\boldsymbol{A},\boldsymbol{B},\boldsymbol{A}_{\rm eq},\boldsymbol{B}_{\rm eq},\boldsymbol{x}_m,\boldsymbol{x}_M,\boldsymbol{x}_0,\texttt{OPT},p_1,p_2,\cdots)$

其中，$\boldsymbol{f}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{A}_{\rm eq}$, $\boldsymbol{B}_{\rm eq}$, $\boldsymbol{x}_m$, $\boldsymbol{x}_M$ 与前面约束与目标函数公式中的记号是完全一致的，$\boldsymbol{x}_0$ 为初始搜索点。各个矩阵约束如果不存在，则应该用空矩阵来占位。`OPT` 为控制

选项，该函数还允许使用附加参数 $p_1, p_2, \cdots$。最优化运算完成后，结果将在变量 $\boldsymbol{x}$ 中返回，最优化的目标函数将在 $f_{\rm opt}$ 变量中返回。我们将通过下面的例子来演示线性规划的求解问题。

【例 6-16】试求解下面的线性规划问题。

$$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} 2x_2+x_3+4x_4+2x_5 \leqslant 54 \\ 3x_1+4x_2+5x_3-x_4-x_5 \leqslant 62 \\ x_1,x_2 \geqslant 0,,x_3 \geqslant 3.32, x_4 \geqslant 0.678, x_5 \geqslant 2.57 \end{cases}} (-2x_1 - x_2 - 4x_3 - 3x_4 - x_5)$$

【求解】从给出的数学式子可以看出，其目标函数可以用其系数向量 $\boldsymbol{f}=[-2,-1,-4,-3,-1]^{\rm T}$ 表示，不等式约束有两个，即

$$\boldsymbol{A}=\begin{bmatrix} 0 & 2 & 1 & 4 & 2 \\ 3 & 4 & 5 & -1 & -1 \end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix} 54 \\ 62 \end{bmatrix}$$

另外，由于没有等式约束，故可以定义 $\boldsymbol{A}_{\rm eq}$ 和 $\boldsymbol{B}_{\rm eq}$ 为空矩阵。由给出的数学问题还可以看出，$\boldsymbol{x}$ 的下界可以定义为 $\boldsymbol{x}_m = [0, 0, 3.32, 0.678, 2.57]^{\rm T}$，且对上界没有限制，故可以将其写成空矩阵。由前面的分析，可以给出如下的 MATLAB 命令来求解线性规划问题，并立即得出结果。

```
>> f=-[2 1 4 3 1]'; A=[0 2 1 4 2; 3 4 5 -1 -1];
   B=[54; 62]; Ae=[]; Be=[]; xm=[0,0,3.32,0.678,2.57];
   ff=optimset; ff.LargeScale='off'; % 不使用大规模问题求解
   ff.TolX=1e-15; ff.TolFun=1e-20; ff.TolCon=1e-20; ff.Display='iter';
   [x,f_opt,key,c]=linprog(f,A,B,Ae,Be,xm,[],[],ff)
Optimization terminated successfully.
x =                        f_opt =                    key =
  19.78500000000000          -89.57499999999999              1
   0.00000000000000
   3.32000000000000
  11.38500000000000
   2.57000000000000
c =
      iterations: 5
       algorithm: 'medium-scale: active-set'
    cgiterations: []
```

从列出的结果看，由于 key 值为 1，故求解是成功的。以上只用了 5 步就得出了线性规划问题的解，可见求解程序功能是很强大的，可以很容易得出线性规划问题的解。

【例 6-17】考虑下面的 4 元线性规划问题，试用 MATLAB 的最优化工具箱求解此问题。

$$\max_{\boldsymbol{x} \text{ s.t.} \begin{cases} x_1/4-60x_2-x_3/50+9x_4 \leqslant 0 \\ -x_1/2+90x_2+x_5/50-3x_4 \geqslant 0 \\ x_3 \leqslant 1, x_1 \geqslant -5, x_2 \geqslant -5, x_3 \geqslant -5, x_4 \geqslant -5 \end{cases}} \left[\frac{3}{4}x_1 - 150x_2 + \frac{1}{50}x_3 - 6x_4\right]$$

【求解】 原问题中应该求解的是最大值问题，所以需要首先将之转换成最小化问题，即将原目标函数乘以 -1，则目标函数将改写成 $-3x_1/4+150x_2-x_3/50+6x_4$。套用线性规划的格式可以得出 $\boldsymbol{f}^{\mathrm{T}}$ 向量为 $[-3/4,150,-1/50,6]$。

再分析约束条件，可见，由最后一条可以写成 $x_i \geqslant -5$，所以可确定自变量的最小值向量为 $\boldsymbol{x}_m$=[-5;-5;-5;-5]。类似地，还能写出自变量的最大值向量为 $\boldsymbol{x}_M$=[Inf;Inf;1;Inf]，其中可以使用 Inf 表示 $+\infty$。约束条件的前两条均为不等式约束，其中第 2 条为 $\geqslant$ 表示，需要将两端均乘以 -1，转换成 $\leqslant$ 不等式，这样可以写出不等式约束为

$$\boldsymbol{A}=\begin{bmatrix}1/4 & -60 & -1/50 & 9\\ 1/2 & -90 & -1/50 & 3\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}0\\0\end{bmatrix}$$

由于原问题中没有等式约束，故应该令 $\boldsymbol{A}_{\mathrm{eq}}$=[]，$\boldsymbol{B}_{\mathrm{eq}}$=[]。最终可以输入如下的命令来求解此最优化问题，得出原问题的最优解。

```
>> f=[-3/4,150,-1/50,6]; Aeq=[]; Beq=[];
   A=[1/4,-60,-1/50,9; 1/2,-90,-1/50,3]; B=[0;0];
   xm=[-5;-5;-5;-5]; xM=[Inf;Inf;1;Inf]; ff=optimset;
   ff.TolX=1e-15; ff.TolFun=1e-20; TolCon=1e-20; ff.Display='iter';
   [x,f_opt,key,c]=linprog(f,A,B,Aeq,Beq,xm,xM,[0;0;0;0],ff)
  Residuals:  Primal     Dual        Upper      Duality     Total
              Infeas     Infeas      Bounds     Gap         Rel
              A*x-b      A'*y+z-w-f  {x}+s-ub   x'*z+s'*w   Error
  ---------------------------------------------------------------
  Iter    0:  9.39e+003 1.43e+002 1.94e+002 6.03e+004 2.77e+001
  Iter    1:  6.38e-012 1.21e+001 0.00e+000 3.50e+003 1.78e+000
  Iter    2:  4.09e-011 5.01e-001 0.00e+000 3.61e+002 3.79e-001
  Iter    3:  3.10e-011 2.24e-001 0.00e+000 3.77e+002 3.62e-001
  ....    ..  ......... ......... ......... ......... .........
  Iter   10:  0.00e+000 6.15e-026 0.00e+000 1.70e-024 4.10e-028
Optimization terminated successfully.
x =                        f_opt =                  key =
  -5.00000000000000          -55.46999999999999          1
  -0.19466666666667
   1.00000000000000
  -5.00000000000000
c =
      iterations: 10
    cgiterations: 0
       algorithm: 'large-scale: interior point'
```

可见，经过 10 步迭代，就能以很高精度得出原问题的最优解。

6.3.3 二次型规划的求解

二次型规划问题是另一种简单的有约束最优化问题，其目标函数为 $\boldsymbol{x}$ 的二次型形式，约束条件仍然为线性不定式约束。一般二次型规划问题的数学表示为

$$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} \boldsymbol{A}\boldsymbol{x}\leqslant\boldsymbol{B} \\ \boldsymbol{A}_{\rm eq}\boldsymbol{x}=\boldsymbol{B}_{\rm eq} \\ \boldsymbol{x}_m\leqslant\boldsymbol{x}\leqslant\boldsymbol{x}_M \end{cases}} \left(\frac{1}{2}\boldsymbol{x}^{\rm T}\boldsymbol{H}\boldsymbol{x}+\boldsymbol{f}^{\rm T}\boldsymbol{x}\right) \tag{6-3-3}$$

和线性规划问题相比，二次型规划目标函数中多了一个二次项 $\boldsymbol{x}^{\rm T}\boldsymbol{H}\boldsymbol{x}$ 来描述 x_i^2 和 x_ix_j 项。MATLAB 的最优化工具箱提供了求解二次型规划问题的 quadprog() 函数，该函数的调用格式为

$[\boldsymbol{x}, f_{\rm opt}$, flag, c]=quadprog($\boldsymbol{H}, \boldsymbol{f}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{A}_{\rm eq}, \boldsymbol{B}_{\rm eq}, \boldsymbol{x}_m, \boldsymbol{x}_M, \boldsymbol{x}_0$, OPT, $p_1, p_2, \cdots$)

其中，函数调用时，$\boldsymbol{H}$ 为二次型规划目标函数中的 $\boldsymbol{H}$ 矩阵，其余各个变量与线性规划函数调用的完全一致。

【例 6-18】 试求解下面的四元二次型规划问题。

$$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} x_1+x_2+x_3+x_4\leqslant 5 \\ 3x_1+3x_2+2x_3+x_4\leqslant 10 \\ x_1,x_2,x_3,x_4\geqslant 0 \end{cases}} \left[(x_1-1)^2+(x_2-2)^2+(x_3-3)^2+(x_4-4)^2\right]$$

【求解】 首先应该将原始问题写成二次型规划的模式。展开目标函数得

$$\begin{aligned} f(x) &= x_1^2-2x_1+1+x_2^2-4x_2+4+x_3^2-6x_3+9+x_4^2-8x_4+16 \\ &= x_1^2+x_2^2+x_3^2+x_4^2-2x_1-4x_2-6x_3-8x_4+30 \end{aligned}$$

因为目标函数中的常数对最优化结果没有影响，所以可以放心地略去。这样就可以将二次型规划标准型中的 $\boldsymbol{H}$ 矩阵和 $\boldsymbol{f}^{\rm T}$ 向量写为

$$\boldsymbol{H}={\rm diag}([2,2,2,2]),\ \ \boldsymbol{f}^{\rm T}=[-2,-4,-6,-8]$$

从而可以给出下列 MATLAB 命令来求解二次型最优化问题。

```
>> f=[-2,-4,-6,-8]; H=diag([2,2,2,2]);
   OPT=optimset; OPT.LargeScale='off'; % 不使用大规模问题算法
   A=[1,1,1,1; 3,3,2,1]; B=[5;10]; Aeq=[]; Beq=[]; LB=zeros(4,1);
   [x,f_opt]=quadprog(H,f,A,B,Aeq,Beq,LB,[],[],OPT)
Optimization terminated successfully.
x =                          f_opt =
   0.00000000000000             -23.66666666666666
   0.66666666666667
   1.66666666666667
   2.66666666666667
```

套用二次型规划标准型时，一定要注意 $\boldsymbol{H}$ 矩阵的生成，因为在式 (6-3-3) 中有一个 1/2 项，所以在本例中，$\boldsymbol{H}$ 矩阵对角元素是 2，而不是 1。另外，这里得出的目标函数实际上不是原始问题中的最优函数，因为人为地除去了常数项。将得出的结果再补上已经除去了的常数项，则可以求出原问题目标函数的值为 6.3333。

6.3.4 一般非线性规划问题的求解

有约束非线性最优化问题的一般描述为

$$\min_{\boldsymbol{x} \text{ s.t. } \boldsymbol{G}(\boldsymbol{x})\leqslant \boldsymbol{0}} f(\boldsymbol{x}) \tag{6-3-4}$$

其中，$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$。为求解方便，约束条件还可以进一步细化为线性等式约束、线性不等式约束、$\boldsymbol{x}$ 变量的上下界向量，还允许一般非线性函数的等式和不等式约束，这时原规划问题可以改写成

$$\min_{\boldsymbol{x} \text{ s.t. } \begin{cases} \boldsymbol{A}\boldsymbol{x}\leqslant \boldsymbol{B} \\ \boldsymbol{A}_{\mathrm{eq}}\boldsymbol{x}=\boldsymbol{B}_{\mathrm{eq}} \\ \boldsymbol{x}_m\leqslant \boldsymbol{x}\leqslant \boldsymbol{x}_M \\ \boldsymbol{C}(\boldsymbol{x})\leqslant \boldsymbol{0} \\ \boldsymbol{C}_{\mathrm{eq}}(\boldsymbol{x})=\boldsymbol{0} \end{cases}} f(\boldsymbol{x}) \tag{6-3-5}$$

MATLAB 最优化工具箱中提供了一个 `fmincon()` 函数，专门用于求解各种约束下的最优化问题。该函数的调用格式为

$[\boldsymbol{x},f_{\mathrm{opt}},\mathtt{flag},\mathtt{c}]=\mathtt{fmincon}(\mathtt{F},\boldsymbol{x}_0,\boldsymbol{A},\boldsymbol{B},\boldsymbol{A}_{\mathrm{eq}},\boldsymbol{B}_{\mathrm{eq}},\boldsymbol{x}_m,\boldsymbol{x}_M,\mathtt{CF},\mathtt{OPT},p_1,p_2,\cdots)$

其中，`F` 为给目标函数写的 M 函数或 `inline()` 函数，$\boldsymbol{x}_0$ 为初始搜索点。各个矩阵约束如果不存在，则应该用空矩阵来占位。`CF` 为给非线性约束函数写的 M 函数，`OPT` 为控制选项。最优化运算完成后，结果将在变量 $\boldsymbol{x}$ 中返回，最优化的目标函数将在 f_{opt} 变量中返回。和其他优化函数一样，选项 `OPT` 有时是很重要的。

【例 6-19】试求解下面的有约束最优化问题。

$$\min_{\boldsymbol{x} \text{ s.t. } \begin{cases} x_1^2+x_2^2+x_3^2-25=0 \\ 8x_1+14x_2+7x_3-56=0 \\ x_1,x_2,x_3\geqslant 0 \end{cases}} \left[1000-x_1^2-2x_2^2-x_3^2-x_1x_2-x_1x_3\right]$$

【求解】分析给出的最优化问题可以发现，约束条件中含有非线性不等式，故而不能使用二次型规划的方式求解，必须用非线性规划的方式来求解。根据给出的问题可以直接写出目标函数为

```
function y=opt_fun1(x)
y=1000-x(1)*x(1)-2*x(2)*x(2)-x(3)*x(3)-x(1)*x(2)-x(1)*x(3);
```

同时，给出的两个约束条件均为等式约束，所以应该写出非线性约束函数为

```
function [c,ceq]=opt_con1(x)
ceq=[x(1)*x(1)+x(2)*x(2)+x(3)*x(3)-25; 8*x(1)+14*x(2)+7*x(3)-56];
c = [];
```

非线性约束函数返回变量分为 c 和 ceq 两个量，其中，前者为不等式约束的数学描述，后者为非线性等式约束，如果某个约束不存在，则应该将其值赋为空矩阵。这样的约束函数处理比早期版本的工具箱中处理更方便、规范。

描述了给出的非线性等式约束后，则 $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{A}_{\rm eq}, \boldsymbol{B}_{\rm eq}$ 都将为空矩阵了。另外，应该给出搜索初值向量 $\boldsymbol{x}_m = [0,0,0]^{\rm T}$，因此，可以调用 fmincon() 函数求解此约束最优化问题。

```
>> ff=optimset; ff.LargeScale='off'; ff.Display='iter';
   ff.TolFun=1e-30; ff.TolX=1e-15; ff.TolCon=1e-20;
   x0=[1;1;1]; xm=[0;0;0]; xM=[]; A=[]; B=[]; Aeq=[]; Beq=[];
   [x,f_opt,c,d]=fmincon('opt_fun1',x0,A,B,Aeq,Beq,xm,xM,'opt_con1',ff);
                                  max                   Directional First-order
 Iter F-count    f(x)    constraint Step-size derivative optimality    Procedure
   0      4     994          27                                   Infeasible start point
   1     11 955.336        22.9       0.25         -295         18.3 infeasible
   2     16 964.012       5.773          1        0.811         6.26 Hessian modified
 ... ..... ....... ......... .........     ........ ........    ...........
  19    103 961.715 7.105e-015           1 -8.69e-015  5.18e-006 Hessian modified
  20    108 961.715           0          1  8.69e-015    0.00146 Hessian modified
  21    113 961.715           0          1 -4.31e-028  1.27e-006 Hessian modified
 Optimization terminated successfully:
  Magnitude of directional derivative in search direction
   less than 2*options.TolFun and maximum constraint violation
   is less than options.TolCon
 No active inequalities
 >> x,f_opt,kk=d.funcCount
 x =                      f_opt =                    kk =
    3.51212119966089          9.617151721300521e+002       113
    0.21698795282036
    3.55217129474683
```

考虑到第 2 个约束条件实际上是线性等式约束，所以能将其从非线性约束函数中除去，故可以将非线性约束函数进一步简化为

```
function [c,ceq]=opt_con2(x)
ceq=x(1)*x(1)+x(2)*x(2)+x(3)*x(3)-25; c = [];
```

线性等式约束可以由相应的矩阵定义出来，这时可以用下面的命令求解原始的最优化问题，且可以得出和前面完全一致的结果。

```
>> x0=[1;1;1]; Aeq=[8,14,7]; Beq=56;
   [x,f_opt,c,d]=fmincon('opt_fun1',x0,A,B,Aeq,Beq,xm,xM,'opt_con2',ff);
   x, f_opt, kk=d.funcCount    % 从略
```

【例 6-20】考虑例 6-19 中给出的最优化问题，试利用梯度求解最优化问题，并比较和原例子中所

用方法的优劣。

【求解】 由给出的目标函数 $f(x)$ 可以立即求出下面的梯度函数(或 Jocobi 矩阵)。

```
>> syms x1 x2 x3; f=1000-x1*x1-2*x2*x2-x3*x3-x1*x2-x1*x3;
   J=jacobian(f,[x1,x2,x3])
J =
   [ -2*x1-x2-x3,     -4*x2-x1,     -2*x3-x1]
```

其数学形式可以写成

$$J=\left[\frac{\partial f}{\partial x_1},\frac{\partial f}{\partial x_2},\frac{\partial f}{\partial x_3}\right]^{\mathrm{T}}=\begin{bmatrix}-2x_1-x_2-x_3\\-4x_2-x_1\\-2x_3-x_1\end{bmatrix}$$

有了梯度,则可以重新改写目标函数为

```
function [y,Gy]=opt_fun2(x)
y=1000-x(1)*x(1)-2*x(2)*x(2)-x(3)*x(3)-x(1)*x(2)-x(1)*x(3);
Gy=-[2*x(1)+x(2)+x(3); 4*x(2)+x(1); 2*x(3)+x(1)];
```

其中,Gy 表示目标函数的梯度向量。再调用最优化求解函数将得出下面的结果:

```
>> x0=[1;1;1]; xm=[0;0;0]; xM=[]; A=[]; B=[]; Aeq=[]; Beq=[];
   ff=optimset; ff.GradObj='on'; ff.Display='iter'; ff.LargeScale='off';
   ff.TolFun=1e-30; ff.TolX=1e-15; ff.TolCon=1e-20;
   [x,f_opt,c,d]=fmincon('opt_fun2',x0,A,B,Aeq,Beq,xm,xM,'opt_con1',ff);
                              max          Directional First-order
 Iter F-count   f(x)   constraint Step-size derivative optimality     Procedure
   0       4     994         27                                  Infeasible start point
   1      11 955.336       22.9      0.25        -295        18.3 infeasible
   2      16 964.012      5.773         1       0.811        6.26 Hessian modified
  ... ..... .......   ......... .........    ........    ........    ...........
  14      76 961.715          0         1   4.35e-015   2.85e-007 Hessian modified
  15      81 961.715          0         1  -1.06e-021   7.59e-008 Hessian modified
  16      86 961.715          0         1  -4.21e-031   3.48e-008 Hessian modified
Optimization terminated successfully:
 Magnitude of directional derivative in search direction
  less than 2*options.TolFun and maximum constraint violation
  is less than options.TolCon
No active inequalities
>> x,f_opt,kk=d.funcCount
x =                         f_opt =                       kk =
   3.51212133105314          9.617151721300522e+002          86
   0.21698794237547
   3.55217116547403
```

可见,若已知目标函数的偏导数,则仅需 86 步目标函数的调用就求出原问题的解,比前面

需要的步数 (113 步) 明显减少。但考虑求取和编写梯度函数所需的时间，实际需要的时间可能更多。注意，若已知梯度函数，则应该将 GradObj 选项设置成 'on'，否则不能识别该梯度。

6.4　整数规划问题的计算机求解

在很多应用领域中，最优化问题的要求除了前面的满足约束条件的规则外，还需要使得全部和部分自变量取整数，这类问题又称为整数规划。其中，要求全部自变量均为整数的最优化问题又称为纯整数规划问题，部分自变量要求为整数的最优化问题又称为混合整数规划的问题。还有一类最优化问题，要求自变量只能是 0 或 1，这类规划问题又称为 0-1 规划问题。整数规划当然又分为线性整数规划问题和非线性整数规划问题。

6.4.1　整数线性规划问题的求解

整数线性规划的一般数学描述为

$$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} \boldsymbol{A}\boldsymbol{x}\leqslant\boldsymbol{B} \\ \boldsymbol{A}_{\rm eq}\boldsymbol{x}=\boldsymbol{B}_{\rm eq} \\ \boldsymbol{x}_m\leqslant\boldsymbol{x}\leqslant\boldsymbol{x}_M \\ \boldsymbol{C}(\boldsymbol{x})\leqslant\boldsymbol{0} \\ \boldsymbol{C}_{\rm eq}(\boldsymbol{x})=\boldsymbol{0} \\ \hat{\boldsymbol{x}}\ \text{为整数} \end{cases}} \boldsymbol{f}^{\rm T}\boldsymbol{x} \tag{6-4-1}$$

其中 $\hat{\boldsymbol{x}}$ 为变量 $\boldsymbol{x}$ 的子集。如果已知自变量所在的区间，则理论上可以考虑用穷举方法列举出区间内所有的变量组合，逐个判定约束条件是否满足，从满足的组合中逐个求取函数的值并排序，由其最小值的对应关系可以简单地求解所需的自变量值。这个方法看似简单、直观，但对稍微多些自变量的情形是不可行的，因为这时计算量为天文数字。相应的数学问题又称为 NP 难 (non-polynomial hard) 问题，故穷举方法只适合于极有限的小规模问题。

MATLAB 自身没有提供整数线性规划的函数，但可以使用荷兰 Eindhoven 科技大学 Michel Berkelaar 等人开发的 LP_Solve 包中的 MATLAB 支持的 mex 文件。该程序据称可以求解多达 30000 个变量，50000 个约束条件的整数线性规划问题，该软件可以从网站 ftp://ftp.ics.ele.tue.nl/pub/lp_solve/ 下载。编译后该函数的调用格式为

```
[x,how]=ipslv_mex(A,B,f,intlist,xM,xm,ctype)
```

其中，$\boldsymbol{A},\boldsymbol{B}$ 表示线性等式和不等式约束。和最优化工具箱中提供的函数不同，这里不要求用多个矩阵分别表示等式和不等式，而可以使用这两个矩阵表示等式、大于式和小于式，具体的约束式子由 ctype 变量控制，ctype(i) 的值为正、零与负分别对应 $\boldsymbol{a}_i\boldsymbol{x}\geqslant b_i$，$\boldsymbol{a}_i\boldsymbol{x}=b_i$ 和 $\boldsymbol{a}_i\boldsymbol{x}\leqslant b_i$。intlist 为整数约束标示，其第 i 个分量为 1 则表示需要 x_i 为整数。返回的 how 为得出解 $\boldsymbol{x}$ 的附加说明，其中，how= 0，表示结果为最优，为 2 表示无可行解，其余的值表示求解失败。

【例 6-21】考虑例 6-16 中给出的线性规划问题，如果要求自变量 x_i 均为整数，则原来的问题就变成了整数线性规划问题，试求解该整数规划问题。

【求解】依照 ipslv_mex() 函数的调用格式，可以立即建立起 $\boldsymbol{f}$ 向量，并建立起 $\boldsymbol{A}, \boldsymbol{B}$ 矩阵和相应的 ctype()，并根据需要建立起 $\boldsymbol{x}_m, \boldsymbol{x}_M$ 向量，这样用下面的语句就可以求解出最优化问题。

```
>> f=-[2 1 4 3 1]'; A=[0 2 1 4 2; 3 4 5 -1 -1]; intlist=ones(5,1);
   B=[54; 62]; ctype=[-1; -1]; xm=[0,0,3.32,0.678,2.57]; xM=inf*ones(5,1);
   [res,b]=ipslv_mex(f,A,B,intlist,xM,xm,ctype)  % 因为返回的 b=0，表示优化成功
res =              b =
   19                  0
    0
    4
   10
    5
```

对于小规模问题，可以考虑采用穷举算法。人为假定 $\boldsymbol{x}_M$ 的各个元素均为 20，当然可以采用下面语句就可以逐个求取函数值，得出所需的结果。

```
>> [x1,x2,x3,x4,x5]=ndgrid(1:20,0:20,4:20,1:20,3:20);
   i=find((2*x2+x3+4*x4+2*x5<=54) & (3*x1+4*x2+5*x3-x4-x5<=62));
   f=-2*x1(i)-x2(i)-4*x3(i)-3*x4(i)-x5(i); [fmin,ii]=sort(f);
   index=i(ii(1)); x=[x1(index),x2(index),x3(index),x4(index),x5(index)]
x =
   19      0      4      10      5
```

可见，结果和上面的完全一致。然而这里有个问题值得注意。其一，本算法得出的结果是 $x_1 \in [0,20]$ 区间的最小值，但这个概念不能随意拓展到此区间之外，如果想将 20 变为 30，在一般的计算机配置下都实现不了，因为所需内存过大，5 个变量的存储量为 $31^5 \times 5 \times 8/2^{20} = 1092.1\text{MB}$ 空间。所以在求解整数规划时不适合采用穷举算法。其二，除了注意得出的最优解之外，事实上还可以得出若干组合，使得该规划函数有次最优解。可以显示排序后的函数值如下：

```
>> fmin(1:10)'
ans =
   -89   -88   -88   -88   -88   -88   -88   -88   -87   -87
>> in=i(ii(1:15)); x=[x1(in),x2(in),x3(in),x4(in),x5(in),fmin(1:15)]
```

可见，函数的最小值为 −89。此外，还有若干个点的值为 −88，求出最优解的同时，还可以列出各个变量的次最优解，如表 6-2 所示。

如果要求 x_1, x_4, x_5 为整数，其他两个变量为任意数，则原问题就变成了混合整数规划问题。用户应该修改一下 intlist 变量，将其设置为 intlist=[1,0,0,1,1]，则可以用下面的语句求出原问题的解。

```
>> intlist=[1,0,0,1,1];
   [res,b]=ipslv_mex(f,A,B,intlist,xM,xm,ctype)
res =                  b =
```

表 6-2 最优解及部分次最优解

x_1	x_2	x_3	x_4	x_5	f	说明	x_1	x_2	x_3	x_4	x_5	f	说明	x_1	x_2	x_3	x_4	x_5	f	说明
19	0	4	10	5	-89	最优	19	0	4	9	7	-88	次优	11	0	8	10	3	-87	次优
18	0	4	11	3	-88	次优	16	0	6	8	8	-88	次优	10	0	9	9	4	-87	次优
17	0	5	10	4	-88	次优	20	0	4	7	11	-88	次优	8	0	10	9	4	-87	次优
15	0	6	10	4	-88	次优	15	0	6	10	3	-87	次优	5	0	12	8	5	-87	次优
12	0	8	9	5	-88	次优	13	0	7	10	3	-87	次优	18	0	4	10	5	-87	次优

```
   19.000000000000000            0
                    0
    3.800000000000000
   11.000000000000000
    3.000000000000000
```

6.4.2 一般非线性整数规划问题与求解

前面的运算局限于线性规划问题的整数规划研究。在实际应用中经常需要求解非线性整数规划或混合规划问题，该领域中一种常用的算法是分枝定界 (branch and bound) 算法，具体算法在这里不详细介绍，只介绍一个基于该算法编写的现成函数 bnb20()，可以用来求解一般非线性整数规划的问题。该函数是荷兰 Groningen 大学的 Koert Kuipers 编写的，可以从 The MathWorks 网站上直接下载。该函数的调用格式为

$[\mathtt{err},f,\boldsymbol{x}]=\mathtt{bnb20}(\mathtt{fun},\boldsymbol{x}_0,\mathtt{intlist},\boldsymbol{x}_m,\boldsymbol{x}_M,\boldsymbol{A},\boldsymbol{B},\boldsymbol{A}_{\rm eq},\boldsymbol{B}_{\rm eq},\mathtt{CFun})$

其中，调用过程中的大部分输入变量与最优化工具箱函数几乎完全一致，该函数直接调用了最优化工具箱中的 fmincon() 函数，该函数还可以根据需要带附加参数，返回的变量 err 为函数的错误信息字符串，$\boldsymbol{x}$ 和 f 分别为最优解和其函数值。如果正确返回最优解，则 err 字符串为空字符串。该函数尚有不完全之处，因为给出的解往往不是很精确的整数，所以应该在该函数调用结束后给出下面的语句将其化成整数。

```
if length(err)==0, x(intlist==1)=round(x(intlist==1)), end
```

【例 6-22】 试用 bnb20() 函数求解例 6-21 中给出的线性整数规划问题。

【求解】 根据要求，可以编写出如下文件来描述目标函数

```
function f=c6miopt(x)
f=-[2 1 4 3 1]*x;
```

和前面介绍的线性规划问题求解不同，上限变量不能再选择为无穷大，而应该选择为较大的数值，例如均选择为 20000。同样，整数的下界也不能再选择为小数，而应该为整数，故也需要相应地变化，这样用下面的语句求解出的线性整数规划问题与例 6-21 得出的完全一致。

```
>> A=[0 2 1 4 2; 3 4 5 -1 -1]; intlist=ones(5,1); Aeq=[]; Beq=[];
   B=[54; 62]; ctype=[-1; -1];
```

```
xm=[0,0,4,1,3]'; xM=20000*ones(5,1); x0=xm;
[errmsg,f,X]=bnb20('c6miopt',x0,intlist,xm,xM,A,B,Aeq,Beq); X=X'
X =
   19      0      4      10      5
```

仍要求 x_1,x_4,x_5 为整数，其他两个变量为任意值，则应该修改一下 intlist 变量，将其设置为 intlist=[1,0,0,1,1]，则可以用下面的语句求出原问题的解，仍与例 6-21 完全一致。

```
>> intlist=[1,0,0,1,1]';xm=[0,0,3.32,1,3]';
   [errmsg,f,X]=bnb20('c6miopt',x0,intlist,xm,xM,A,B,Aeq,Beq); X
X =
  19.00000000000000
                  0
   3.80000000000000
  11.00000000000000
   3.00000000000000
```

【例 6-23】对著名的 Rosenbrock 函数稍加修改，就可以 $f(\boldsymbol{x})=100(x_2-x_1^2)^2-(4.5543-x_2)^2$，试求解整数 x_1 和 x_2，使得

$$\min_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}-100\leqslant x_1\leqslant 100\\-100\leqslant x_2\leqslant 100\end{array}\right.} f(\boldsymbol{x})$$

【求解】由给定的目标函数可以编写出如下的 MATLAB 函数：

```
function y=c6fun1(x)
y=100*(x(2)-x(1)^2)^2+(4.5543-x(1))^2;
```

在一般最优化问题中，(1,1) 显然为最优点，这里考虑整数规划问题的求解方法。注意，bnb20() 函数只能调用上面函数形式描述的目标函数，而不能使用 inline() 函数来描述。另外，还能如下给出上下界与约束条件，最后得出原来问题的解。

```
>> x0=[1;1]; xm=-100*[1;1]; xM=100*[1;1];
   A=[]; B=[]; Aeq=[]; Beq=[]; intlist=[1,1]';
   [errmsg,f,x]=bnb20('c6fun1',x0,intlist,xm,xM,A,B,Aeq,Beq); x'
ans =
   5.00000000000000     24.99999981456858
>> if length(errmsg)==0, x=round(x), end; x=x'
x =
     5     25
```

由得出的解可以看出，开始时 x_2 稍有误差，经取整得 $x_1=5, x_2=25$。注意，在这样的问题求解中，搜索区域的选择也是很重要的，如果用户为节省时间选择了一个更小的区间，如 $x_{1,2}\in[-20,20]$，则将得出如下结果，可见该值不是原问题的最优解。

```
>> xm=-20*[1;1]; xM=20*[1;1];
   [errmsg,f,x]=bnb20('c6fun1',x0,intlist,xm,xM,A,B,Aeq,Beq);
```

```
  if length(errmsg)==0, x=round(x), end; x=x';
x =
     4     16
```

其实，对这样小规模的问题，选择较大的搜索范围，如 $x_{1,2} \in (-200,200)$，用穷举搜索算法能立即得出问题的解，和上述结果一致。

```
>> [x1,x2]=meshgrid(-200:200); f=100*(x2-x1.^2).^2+(4.5543-x1).^2;
   [fmin,i]=sort(f(:));  x=[x1(i(1)),x2(i(1))]
x =
     5    25
```

6.4.3 0-1 规划问题求解

所谓 0-1 规划，即指自变量 x_i 的值或者为 0，或者为 1。所以求解 0-1 规划看起来很简单，让每个自变量 x_i遍取 0,1，在得出的组合中选择既满足约束条件又使目标函数取最小值的项。而事实上，随着问题规模的增大，这样的计算量将按指数增长。例如，自变量的个数为 n，则需要执行的循环次数将为 2^n，在 n 较大时其值可能是个天文数字，故仍然需要考虑其他算法进行求解。

MATLAB 7.0 版本提供了一个新函数 bintprog()，可以用来求解 0-1 线性规划问题，但不能直接求解非线性 0-1 规划问题的求解。该函数的调用格式为

$$\boldsymbol{x}=\texttt{bintprog}(\boldsymbol{f},\boldsymbol{A},\boldsymbol{B},\boldsymbol{A}_{\rm eq},\boldsymbol{B}_{\rm eq})$$

该函数可以直接用来求解下面例子演示的 0-1 线性规划问题。

【例 6-24】试求解下面给出的 0-1 线性规划问题。

$$\min_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}x_1+2x_2-x_3\leqslant 2\\ x_1+4x_2+x_3\leqslant 4\\ x_1+x_2\leqslant 3\\ 4x_2+x_3\leqslant 6\end{array}\right.} (-3x_1+2x_2+5x_3)$$

【求解】套用所需的最优化模型，可以立即求出 $\boldsymbol{f}$、$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 矩阵，这样可以给出如下的语句求解 0-1 线性规划问题，得出

```
>> f=[-3,2,-5]; A=[1 2 -1; 1 4 1; 1 1 0; 0 4 1]; B=[2;4;5;6];
   x=bintprog(f,A,B,[],[])'
x =
     1     0     1
```

即，0-1 规划的结果为 $x_1=1, x_2=0, x_3=1$。

对于小规模问题，当然可以采用下面语句，逐个判定约束条件并找找出目标函数的值，通过排序即能得出所需的结果，且可以保证此结果为全局最优解。

```
>> [x1,x2,x3]=meshgrid([0,1]);
   i=find((x1+2*x2-x3<=2) & (x1+4*x2+x3<=4) & (x1+x2<=3) & (4*x1+x3<=6));
   f=-3*x1(i)+2*x2(i)-5*x3(i); [fmin,ii]=sort(f);
```

```
  index=i(ii(1)); x=[x1(index),x2(index),x3(index)]
x =
     1       0       1
>> x=[x1(i(ii)),x2(i(ii)),x3(i(ii))]; [x fmin]  % 还可以列出所有的可行解
ans =
     1     0     1    -8
     0     0     1    -5
     1     0     0    -3
     0     0     0     0
     0     1     0     2
```

调用前面的 ipslv_mex() 或 bnb20() 函数，设定上下限 x_m 和 x_M 分别为零向量和幺向量，这样再求整数规划就能得出原问题的解。

【例 6-25】试用 ipslv_mex() 函数求解例 6-24 给出的 0-1 线性规划问题。

【求解】按照给出的问题，可以设定 x_m 和 x_M 分别为零向量和幺向量，这样可以给出如下的语句求解 0-1 整数规划问题，得出

```
>> f=[-3,2,-5]; xm=[0;0;0]; xM=[1;1;1]; intlist=[1;1;1];
  A=[1 2 -1; 1 4 1; 1 1 0; 0 4 1]; B=[2;4;5;6]; ctype=-1*ones(4,1);
  [res,b]=ipslv_mex(f,A,B,intlist,xM,xm,ctype)
res =                    b =
     1                        0
     0
     1
```

结果和前面得出的完全一致。事实上，分析给定的约束条件，可以发现后两个约束条件是冗余的，可以取消。

由于 MATLAB 及其最优化工具箱对整数规划的支持功能很有限，免费的 MATLAB 函数求解非线性混合整数规划的功能也不是很强大，故 TOMLAB Optimization 公司推出了基于 MATLAB 的第三方软件 TOMLAB[①]，可以更好地求解混合整数规划问题。该软件是商品软件，使用起来也没有 MATLAB 最优化工具箱和前面介绍的免费工具那么简单，但其功能齐全，能够求解各类最优化问题，是当前基于 MATLAB 的最好的最优化工具。限于篇幅，本书不介绍该工具箱。

6.5 本章要点简介

- 本章有关的 MATLAB 函数及自编函数由下表给出。

函数名	函 数 功 能	工具箱	本书页码
solve()	方程的解析解，尤其适用于多项式方程	符号运算	173

[①]该工具箱的下载网址为 http://www.tomlab.com，读者可以下载该工具箱的 21 天免费试用版。

续表

函数名	函 数 功 能	工具箱	本书页码
fsolve()	方程的数值解	MATLAB	176
optimset()	最优化控制参数	最优化	176
fminsearch()	无约束最优化问题求解	MATLAB	181
fminunc()	无约束最优化问题求解	最优化	181
linprog()	线性规划问题求解	最优化	187
quadprog()	二次型规划问题求解	最优化	190
fmincon()	一般非线性规划问题求解	最优化	191
ipslv_mex()	线性混合整数规划问题求解，可以用于 0-1 规划	下载	194
bnb20()	一般整数混合规划问题分枝定界法求解，可以用于 0-1 规划	下载	196
bintprog()	MATLAB 7.0 提供的新 0-1 线性规划求解函数	最优化	198

- 数学方程求解是科学与工程研究中经常遇到的问题。本章先介绍了简单方程的图解法，给出了方程求解的基本概念，并介绍了基于符号运算工具箱中 solve() 函数的多项式类方程的准解析求解算法，还介绍了基于最优化工具箱 fsolve() 函数求取一般非线性方程的数值解法。
- 本章介绍了无约束最优化问题及 MATLAB 解决方案，并引入了全局最优解与局部最优解的概念。
- 本章介绍了若干种有约束最优化问题和可行解区域的概念，还介绍了最优化问题求解算法的 MATLAB 求解方法，如线性规划问题、二次型规划问题及一般非线性规划问题，用这样的方法可以轻易求解出较复杂的非线性规划问题。
- 本章系统地介绍了整数规划问题的计算机求解方法，引入了整数线性规划问题求解工具箱、一般非线性整数规划问题求解函数 bnb20()。
- 由 MATLAB 7.0 版本中给出的新函数 bintprog() 探讨了 0-1 整数线性规划问题的计算机求解。借助分枝定界法的 bnb20() 函数还可以求解 0-1 非线性规划问题。
- 目前，最优化问题最好的工具箱是商品软件 TOMLAB，其功能很强大，然而限于篇幅，本章并未详细介绍该工具。另外，基于遗传算法的最优化方法是当前被认为有可能获得全局最优解的有前途的最优化方法，基于遗传算法的最优化方法及其 MATLAB 实现问题将在第 10.3 节中详细介绍。

6.6 习 题

1 求解能转换成多项式方程的联立方程，并检验得出的高精度数值解的精度。

① $\begin{cases} x_1^2 - x_2 - 1 = 0 \\ (x_1 - 2)^2 + (x_2 - 0.5)^2 - 1 = 0 \end{cases}$， ② $\begin{cases} x^2y^2 - zxy - 4x^2yz^2 = xz^2 \\ xy^3 - 2yz^2 = 3x^3z^2 + 4xzy^2 \\ y^2x - 7xy^2 + 3xz^2 = x^4zy \end{cases}$

2 试用图解法求解下面的一元和二元方程，并验证得出的结果。

① $f(x)=\mathrm{e}^{-(x+1)^2+\pi/2}\sin(5x+2)$， ② $f(x,y)=(x^2+y^2+xy)\mathrm{e}^{-x^2-y^2-xy}$

3 用数值求解函数求解习题 2 中方程的根，并对得出的结果进行检验。

4 试求出使得 $\displaystyle\int_0^1(\mathrm{e}^x-cx)^2\,\mathrm{d}x$ 取得极小值的 c 值。

5 试求解下面的无约束最优化问题。

$$\min_{\boldsymbol{x}}\quad \begin{array}{l}100(x_2-x_1^2)^2+(1-x_1)^2+90(x_4-x_3^2)+(1-x_3^2)^2+\\ 10.1\left[(x_2-1)^2+(x_4-1)^2\right]+19.8(x_2-1)(x_4-1)\end{array}$$

6 考虑 Rastrigin 函数[25] $f(x_1,x_2)=20+x_1^2+x_2^2-10(\cos\pi x_1+\cos\pi x_2)$，试用三维曲面绘制该函数的函数值，选择初值求取该函数的最小值，并理解全局最优解和局部最优解的概念以及最优解对初值的依赖关系。

7 试用图解法求解下面的非线性规划问题，并用数值求解算法验证结果。

$$\min_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}x_1-x_2+2\geqslant0\\ -x_1^2+x_2-1\geqslant0\\ x_1\geqslant0,x_2\geqslant0\end{array}\right.}\quad(x_1^3+x_2^2-4x_1+4)$$

8 试求解下面的线性规划问题。

① $$\min_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}4x_1-x_2+2x_3-x_4=-2\\ x_1+x_2-x_3+2x_4\leqslant14\\ 2x_1-3x_2-x_3-x_4\geqslant-2\\ x_{1,2,3}\geqslant-1,x_4\ \text{无约束}\end{array}\right.}\quad -3x_1+4x_2-2x_3+5x_4$$

② $$\min_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}x_1+x_2+x_3+x_4=4\\ -2x_1+x_2-x_3-x_6+x_7=1\\ 3x_2+x_3+x_5+x_7=9\\ x_{1,2,\cdots,7}\geqslant0\end{array}\right.}\quad x_6+x_7$$

③ $$\max_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}x_{11}+x_{12}+x_{13}+x_{14}\geqslant15\\ x_{12}+x_{13}+x_{14}+x_{21}+x_{22}+x_{23}\geqslant10\\ x_{13}+x_{14}+x_{22}+x_{23}+x_{31}+x_{32}\geqslant20\\ x_{14}+x_{23}+x_{32}+x_{41}\geqslant12\\ x_{ij}\geqslant0,\ (i=1,2,3,4,j=1,2,3,4)\end{array}\right.}\quad 2800(x_{11}+x_{21}+x_{31}+x_{41})+4500(x_{12}+x_{22}+x_{32})+6000(x_{13}+x_{23})+7300x_{14}$$

9 试求解下面的二次型规划问题，并用图示的形式解释结果。

① $$\min_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}x_1+x_2\leqslant3\\ 4x_1+x_2\leqslant9\\ x_{1,2}\geqslant0\end{array}\right.}\quad 2x_1^2-4x_1x_2+4x_2^2-6x_1-3x_2$$

② $$\min_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}-x_1+x_2=1\\ x_1+x_2\leqslant2\\ x_{1,2}\geqslant0\end{array}\right.}\quad (x_1-1)^2+(x_2-2)^2$$

10 试求解下面的非线性规划问题。

① $$\min_{\boldsymbol{x}\ \text{s.t.}\left\{\begin{array}{l}x_1+x_2\leqslant0\\ -x_1x_2+x_1+x_2\geqslant1.5\\ x_1x_2\geqslant-10\\ -10\leqslant x_1,x_2\leqslant10\end{array}\right.}\quad \mathrm{e}^{x_1}(4x_1^2+2x_2^2+4x_1x_2+2x_2+1)$$

② $$\max_{\boldsymbol{x} \text{ s.t.} \begin{cases} 0.003079x_1^3x_2^3x_5-\cos^3 x_6\geqslant 0 \\ 0.1017x_3^3x_4^3-x_5^2\cos^3 x_6\geqslant 0 \\ 0.09939(1+x_5)x_1^3x_2^2-\cos^2 x_6\geqslant 0 \\ 0.1076(31.5+x_5)x_3^3x_4^2-x_5^2\cos^2 x_6\geqslant 0 \\ x_3x_4(x_5+31.5)-x_5[2(x_1+5)\cos x_6+x_1x_2x_5]\geqslant 0 \\ 0.2\leqslant x_1\leqslant 0.5, 14<\leqslant x_2\leqslant 22, 0.35\leqslant x_3\leqslant 0.6, \\ 16\leqslant x_4\leqslant 22, 5.8\leqslant x_5\leqslant 6.5, 0.14\leqslant x_6\leqslant 0.2618 \end{cases}} \frac{1}{2\cos x_6}\left[x_1x_2(1+x_5)+x_3x_4\left(1+\frac{31.5}{x_5}\right)\right]$$

11 求解下面的整数线性规划问题。

① $$\max_{\boldsymbol{x} \text{ s.t.} \begin{cases} \boldsymbol{x}\geqslant 0 \\ 3534x_1+2356x_2+1767x_3+589x_4+528x_5+451x_6+304x_7\leqslant 119567 \end{cases}} (592x_1+381x_2+273x_3+55x_4+48x_5+37x_6+23x_7)$$

② $$\max_{\boldsymbol{x} \text{ s.t.} \begin{cases} x_1+x_2+x_3=30 \\ x_4+x_5+x_6=18 \\ x_1+x_4=10 \\ x_2+x_5\leqslant 18 \\ x_3+x_6\geqslant 30 \\ x_{1,\cdots,6}\geqslant 0 \end{cases}} (120x_1+66x_2+72x_3+58x_4+132x_5+104x_6)$$

12 试求解下面的 0-1 线性规划问题，并对①、②题用穷举方法检验得出的结果。

① $$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} x_1-x_2+5x_3+x_4-4x_5\geqslant 2 \\ -2x_1+6x_2-3x_3-2x_4+2x_5\geqslant 0 \\ -2x_2+2x_3-x_4-x_5\leqslant 1 \\ 0\leqslant x_i\leqslant 1 \end{cases}} (5x_1+7x_2+10x_3+3x_4+x_5)$$

② $$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} x_1-x_6\leqslant 0 \\ x_1-x_5\leqslant 0 \\ x_2-x_4\leqslant 0 \\ x_2-x_5\leqslant 0 \\ x_3-x_4\leqslant 0 \\ x_1+x_2+x_3\leqslant 2 \\ 0\leqslant x_i\leqslant 1 \end{cases}} (-3x_1-4x_2-5x_3+4x_4+4x_5+2x_6)$$

③ $$\max \begin{bmatrix} 1898 & 440 & 22507 & 270 & 14148 & 3100 & 4650 & 30800 & 615 & 4975 \\ & 1160 & 4225 & 510 & 11880 & 479 & 440 & 490 & 330 & 110 \\ & 560 & 24355 & 2885 & 11748 & 4550 & 750 & 3720 & 1950 & 10500 \end{bmatrix} \boldsymbol{x}$$

$$\boldsymbol{x} \text{ s.t.} \begin{cases} \begin{bmatrix} 45 & 0 & 85 & 150 & 65 & 95 & 30 & 0 & 170 & 0 & 40 & 25 & 20 & 0 & 0 & 25 & 0 & 0 & 25 & 0 & 165 & 0 & 85 & 0 & 0 & 0 & 0 & 100 \\ 30 & 20 & 125 & 5 & 80 & 25 & 35 & 73 & 12 & 15 & 15 & 40 & 5 & 10 & 10 & 12 & 10 & 9 & 0 & 20 & 60 & 40 & 50 & 36 & 49 & 40 & 19 & 150 \end{bmatrix} \boldsymbol{x} \leqslant \begin{bmatrix} 600 \\ 600 \end{bmatrix} \\ 0\leqslant x_i\leqslant 1 \end{cases}$$

说明，③题中目标函数的系数实际上是行向量，由于排版原因只能这样表示。

第7章　微分方程问题的计算机求解

微分方程是描述动态系统的最常用数学工具，也是很多科学与工程领域数学建模的基础。线性微分方程和低阶特殊微分方程往往可以通过解析解的方法求解，但一般的非线性微分方程是没有解析解的，故需要用数值解的方式求解。第 7.1 节将研究微分方程的解析解算法，介绍在 MATLAB 环境中如何用微分方程求解函数直接得出线性微分方程组的解析解，并对一阶简单的非线性微分方程的解析解求解进行探讨，从而并得出结论，一般非线性微分方程是没有解析解的。第 7.2 节将介绍一般的一阶显式常微分方程组的数值求解方法及 MATLAB 实现，并将介绍其他类型常微分方程组转换成一阶显式常微分方程组的方法。第 7.3 节将介绍其他各类常微分方程数值求解的方法及 MATLAB 实现，包括刚性微分方程、微分代数方程组、隐式微分方程组以及延迟微分方程组的求解。第 7.4 节和第 7.5 节将分别介绍常微分方程组边值问题的数值解法和一般偏微分方程的数值求解方法，并将介绍如何用 MATLAB 语言的偏微分方程工具箱提供的程序界面求解偏微分方程的方法及步骤。第 7.6 节还将简介 Simulink 仿真环境，并将介绍如何在 Simulink 环境下建立微分方程的数学模型，还将通过仿真方法求解微分方程的一般步骤及方法，用这样的方法理论上可以求解任意复杂的常微分方程组初值问题数值解。

7.1　常系数线性微分方程的解析解方法

7.1.1　线性常系数微分方程解析解的数学描述

假设已知常系数线性微分方程的一般描述方法为

$$\begin{aligned}\frac{\mathrm{d}^n y(t)}{\mathrm{d}t^n}+a_1\frac{\mathrm{d}^{n-1}y(t)}{\mathrm{d}t^{n-1}}+a_2\frac{\mathrm{d}^{n-2}y(t)}{\mathrm{d}t^{n-2}}+\cdots+a_{n-1}\frac{\mathrm{d}y(t)}{\mathrm{d}t}+a_n y(t)=\\ b_1\frac{\mathrm{d}^m u(t)}{\mathrm{d}t^m}+b_2\frac{\mathrm{d}^{m-1}u(t)}{\mathrm{d}t^{m-1}}+\cdots+b_m\frac{\mathrm{d}u(t)}{\mathrm{d}t}+b_{m+1}u(t)\end{aligned} \tag{7-1-1}$$

其中，a_i, b_i 均为常数，利用第 5.1 节介绍的性质，对零初值问题有 $\mathscr{L}[\mathrm{d}^m y(t)/\mathrm{d}t^m]=s^m\mathscr{L}[y(t)]$，可以对应得出下面的多项式代数方程

$$s^n+a_1s^{n-1}+a_2s^{n-2}+\cdots+a_{n-1}s+a_n=0 \tag{7-1-2}$$

假设代数方程的特征根 s_i 均可以求出，且假设它们均相异，则可以得出原微分方程的解析解一般形式为

$$y(t)=C_1\mathrm{e}^{r_1t}+C_2\mathrm{e}^{r_2t}+\cdots+C_n\mathrm{e}^{r_nt}+\gamma(t) \tag{7-1-3}$$

其中，C_i 为待定系数，而 $\gamma(t)$ 是满足 $u(t)$ 输入的一个特解。s_i 有重根的情况也有相应的解析解形式。

从得出的代数方程 (7-1-2) 看，由著名的 Abel 定理可知，4 次及以下的多项式代数方程是能求出根的解析解的，故可以得出结论，低次阶线性微分方程有一般意义下的解析解，结合多项式方程的数值解法可以得出一般高次多项式代数方程的数值解法，即高阶线性微分方程的准解析解方法。本节将介绍用 MATLAB 语言及其符号运算工具箱求解线性常系数微分方程解析解的方法。

7.1.2 微分方程的解析解方法

MATLAB 语言的符号运算工具箱提供了一个线性常系数微分方程求解的实用函数 dsolve()，该函数允许用字符串的形式描述微分方程及初值、边值条件，最终将得出微分方程的解析解。该函数的调用格式为

```
y=dsolve(f1, f2, ⋯, fm)
y=dsolve(f1, f2, ⋯, fm,'x')     指明自变量
```

其中，f_i 既可以描述微分方程，又可以描述初始条件或边界条件。在描述微分方程时，可以用 D4y 这样的记号表示 $y^{(4)}(t)$，还可以用 D2y(2)=3 这类记号表示 $\ddot{y}(2)=3$ 这样的已知条件，该函数可以容易地得出原微分方程的解。如果描述微分方程的自变量不是 t 而是 x，则可以由后一个 MATLAB 语句格式指明自变量。

【例 7-1】假设输入信号为 $u(t)=\mathrm{e}^{-5t}\cos(2t+1)+5$，试求出下面微分方程的通解。

$$y^{(4)}(t)+10y^{(3)}(t)+35\ddot{y}(t)+50\dot{y}(t)+24y(t)=5\ddot{u}(t)+4\dot{u}(t)+2u(t)$$

【求解】若想求解本微分方程，首先应该定义 t 为符号变量，这样就可以推导出给定微分方程等式右侧的时间表达式为

```
>> syms t; u=exp(-5*t)*cos(2*t+1)+5;
   uu=5*diff(u,t,2)+4*diff(u,t)+2*u
uu =
   87*exp(-5*t)*cos(2*t+1)+92*exp(-5*t)*sin(2*t+1)+10
```

这样，原微分方程的通解可以通过下面的语句直接求出。

```
>> syms t y;
   y=dsolve(['D4y+10*D3y+35*D2y+50*Dy+24*y=',...
      '87*exp(-5*t)*cos(2*t+1)+92*exp(-5*t)*sin(2*t+1)+10'])
y =
   5/12-343/520*exp(-5*t)*cos(2*t+1)-547/520*exp(-5*t)*sin(2*t+1)+
      C1*exp(-4*t)+C2*exp(-3*t)+C3*exp(-2*t)+C4*exp(-t)
```

该结果用 LaTeX 可以得出更好的显示为

$$y(t)=\frac{5}{12}-\frac{343}{520}\mathrm{e}^{-5t}\cos(2t+1)-\frac{547}{520}\mathrm{e}^{-5t}\sin(2t+1)+C_1\mathrm{e}^{-4t}+C_2\mathrm{e}^{-3t}+C_3\mathrm{e}^{-2t}+C_4\mathrm{e}^{-t}$$

其中，C_i 为任意常数。若给出初始条件或边界条件，则可以通过这些条件建立方程，求出 C_i 的值。这样的结果和高等数学中微分方程求解是一致的。

仍考虑上面的微分方程，假设已知 $y(0)=3, \dot{y}(0)=2, \ddot{y}(0)=y^{(3)}(0)=0$，则可以通过下面的命令求取满足该微分方程的特解。

```
>> y=dsolve(['D4y+10*D3y+35*D2y+50*Dy+24*y=',...
     '87*exp(-5*t)*cos(2*t+1)+92*exp(-5*t)*sin(2*t+1)+10'],'y(0)=3',...
     'Dy(0)=2','D2y(0)=0','D3y(0)=0')
```

由于得出的解较冗长，所以这里只给出自动转换的 LaTeX 结果如下：

$$y(t)=\frac{5}{12}-\frac{343}{520}\mathrm{e}^{-5t}\cos(2t+1)-\frac{547}{520}\mathrm{e}^{-5t}\sin(2t+1)+\left(-\frac{445}{26}\cos 1-\frac{51}{13}\sin 1-\frac{69}{2}\right)\mathrm{e}^{-2t}+\left(-\frac{271}{30}\cos 1+\frac{41}{15}\sin 1-\frac{25}{4}\right)\mathrm{e}^{-4t}+\left(\frac{179}{8}\cos 1+\frac{5}{8}\sin 1+\frac{73}{3}\right)\mathrm{e}^{-3t}+\left(\frac{133}{30}\cos 1+\frac{97}{60}\sin 1+19\right)\mathrm{e}^{-t}$$

由得出的结果还可以看出，$\mathrm{e}^{-\alpha t}$ 的系数均是常数，采用数值算法逐项处理其系数如下：

```
>> [n,d]=rat(double(vpa(-445/26*cos(1)-51/13*sin(1)-69/2))); [n,d]
ans =
        -8704          185
```

则可以得出一组有理式近似结果，最终可以近似写出方程解的解析形式为

$$y(t)\simeq\frac{5}{12}-\frac{343}{520}\mathrm{e}^{-5t}\cos(2t+1)-\frac{547}{520}\mathrm{e}^{-5t}\sin(2t+1)-\frac{8704}{185}\mathrm{e}^{-2t}-\frac{2243}{254}\mathrm{e}^{-4t}+\frac{5025}{136}\mathrm{e}^{-3t}+\frac{2981}{131}\mathrm{e}^{-t}$$

事实上，这样的结果是对原始系数的近似，因为实际系数不是真正的有理数。该系数的实际误差可以由下面的语句进行估算，得出较小的误差。

```
>> vpa(-445/26*cos(sym(1))-51/13*sin(1)-69/2+8704/185)
ans =
   .11473197586479092256414463 6e-4
```

利用强大的 MATLAB 符号运算工具箱，还可以求解出以前看似不可能的问题的解析解。例如，设置 $y(0)=1/2, \dot{y}(\pi)=1, \ddot{y}(2\pi)=0, \dot{y}(2\pi)=1/5$，则可以得出解析解为

```
>> y=dsolve(['D4y+10*D3y+35*D2y+50*Dy+24*y=',...
     '87*exp(-5*t)*cos(2*t+1)+92*exp(-5*t)*sin(2*t+1)+10'],'y(0)=1/2',...
     'Dy(pi)=1','D2y(2*pi)=0','Dy(2*pi)=1/5')
```

如果用推导的方法求 C_i 的值，则每个系数的解析解至少要写出 10 数行，所以应该采用有理式近似的方式将方程的解析解表示成[①]

```
>> vpa(y,10)
ans =
5/12-343/520*exp(-5t)*cos(2t+1)-547/520*exp(-5t)*sin(2t+1)-219.1291604*exp(-t)
  +442590.9052*exp(-4.*t)+31319.63786*exp(-2.*t)-473690.0889*exp(-3.*t)
```

【例 7-2】 前面介绍的方程只含有实数极点，其实符号运算工具箱提供的 dsolve() 函数同样适用

[①] 在不影响解析解意义的前提下，为排版效果起见这里已对解析解的表示形式稍作修改。

于有复数极点的微分方程解析解。假设微分方程如下

$$y^{(5)}(t)+5y^{(4)}(t)+12y^{(3)}(t)+16\ddot{y}(t)+12\dot{y}(t)+4y(t)=\dot{u}(t)+3u(t)$$

且假设输入信号为正弦信号 $u(t)=\sin t$，并假设 $y(0)=\dot{y}(0)=\ddot{y}(0)=y^{(3)}(0)=y^{(4)}(0)=0$，试用解析方法求解该方程。

【求解】 用下面的方法可以求出原微分方程的解析解

```
>> syms t y; u=sin(t); uu=3*diff(u)+3*u
uu =
   3*cos(t)+3*sin(t)
>> y=dsolve('D5y+5*D4y+12*D3y+16*D2y+12*Dy+4*y=3*cos(t)+3*sin(t)',...
      'y(0)=0','Dy(0)=0','D2y(0)=0','D3y(0)=0','D4y(0)=0')
y =
    -12/25*cos(t)-9/25*sin(t)+57/50*exp(-t)*sin(t)+
      12/25*exp(-t)*cos(t)+3/5*exp(-t)*sin(t)*t-3/10*exp(-t)*cos(t)*t
```

其解析解的数学描述为

$$y(t)=-\frac{12}{25}\cos t-\frac{9}{25}\sin t+\frac{57}{50}\mathrm{e}^{-t}\sin t+\frac{12}{25}\mathrm{e}^{-t}\cos t+\frac{3}{5}t\mathrm{e}^{-t}\sin t-\frac{3}{10}t\mathrm{e}^{-t}\cos t$$

或更简单地，手动修改为

$$y(t)=-\frac{12}{25}\cos t-\frac{9}{25}\sin t+\left(\frac{57}{50}+\frac{3}{5}t\right)\mathrm{e}^{-t}\sin t+\left(\frac{12}{25}-\frac{3}{10}t\right)\mathrm{e}^{-t}\cos t$$

【例 7-3】 试求解下面的线性微分方程组

$$\begin{cases}\ddot{x}(t)+2\dot{x}(t)=x(t)+2y(t)-\mathrm{e}^{-t}\\ \dot{y}(t)=4x(t)+3y(t)+4\mathrm{e}^{-t}\end{cases}$$

【求解】 线性微分方程组也可以用 dsolve() 函数直接求解。上述的线性微分方程组可以由下面的 MATLAB 语句直接求解。

```
>> [x,y]=dsolve('D2x+2*Dx=x+2*y-exp(-t)','Dy=4*x+3*y+4*exp(-t)')
```

用 LaTeX 处理得出的结果，得

$$\begin{cases}x(t)=-6t\mathrm{e}^{-t}+C_1\mathrm{e}^{-t}+C_2\mathrm{e}^{(1+\sqrt{6})t}+C_3\mathrm{e}^{-(-1+\sqrt{6})t}\\ y(t)=6t\mathrm{e}^{-t}-C_1\mathrm{e}^{-t}+2\left(2+\sqrt{6}\right)C_2\mathrm{e}^{(1+\sqrt{6})t}+2\left(2-\sqrt{6}\right)C_3\mathrm{e}^{-(-1+\sqrt{6})t}+\frac{1}{2}\mathrm{e}^{-t}\end{cases}$$

7.1.3 Laplace 变换在线性微分方程求解中的应用

重新考虑式 (7-1-1) 中给出的常系数线性微分方程模型，假设输入信号 $u(t)$ 和输出信号 $y(t)$ 及其各阶导数在 $t=0$ 处的值均为 0，则对方程两端均进行 Laplace 变换，记 $Y(s)=\mathscr{L}[y(t)]$, $U(s)=\mathscr{L}[u(t)]$，则 $Y(s)/U(s)$ 可以表示成一个有理函数，在自动控制领域称为系统的传递函数，亦即

$$G(s)=\frac{Y(s)}{U(s)}=\frac{b_1s^m+b_2s^{m-1}+\cdots+b_ms+b_{m+1}}{s^n+a_1s^{n-1}+a_2s^{n-2}+\cdots+a_{n-1}s+a_n} \tag{7-1-4}$$

如果可以将 $U(s)$ 也表示成一个有理函数 (事实上大部分常用输入信号均可以表示成有理函数形式)，则可以将输出信号的 Laplace 变换 $Y(s)$ 写成

$$Y(s)=\frac{\hat{b}_1 s^m+\hat{b}_2 s^{m-1}+\cdots+\hat{b}_m s+\hat{b}_{m+1}}{s^n+\hat{a}_1 s^{n-1}+\hat{a}_2 s^{n-2}+\cdots+\hat{a}_{n-1}s+\hat{a}_n} \tag{7-1-5}$$

这样，原微分方程的解析解可以由 Laplace 反变换求出，即 $y(t)=\mathscr{L}^{-1}[Y(s)]$。前面介绍过，如果 $Y(s)$ 为有理函数，则可以通过 `residue()` 函数求出其部分分式展开式，再由 Laplace 反变换函数 `ilaplace()` 可以求出输出信号的解析解 $y(t)$。如果 $Y(s)$ 的分母多项式有复数根，为得出可读性很高的结果，仍建议用 `pfrac()` 函数对之进行进一步处理，得出可读性更强的结果。

【例 7-4】 试求出例 7-1 中给出的微分方程在输出信号 $y(t)$ 及其各阶导数在 $t=0$ 时刻均为 0 的解析解。为方便起见，这里重新写出原微分方程为

$$y^{(4)}(t)+10y^{(3)}(t)+35\ddot{y}(t)+50\dot{y}(t)+24y(t)=5\ddot{u}(t)+4\dot{u}(t)+2u(t)$$

并假定输入信号为 $u(t)=\mathrm{e}^{-5t}\cos(2t+1)+5$。

【求解】 对此微分方程两端直接求取 Laplace 变换，则很容易得出下面的式子。

$$s^4Y(s)+10s^3Y(s)+35s^2Y(s)+50sY(s)+24Y(s)=5s^2U(s)+4sU(s)+2U(s)$$

由已知的输入信号可以得出 $U(s)$，代入后解出 $Y(s)$，再求取 Laplace 反变换，则可以得出方程的解析解。仔细分析上述的步骤，可以发现该方法是错误的，由微分方程变换成上面的式子应该有前提条件，即 $u(t)$ 及其各阶导数在 $t=0$ 处的值均为 0，而从输入信号看这个条件显然不满足，所以其关键式子有问题。

因为 $u(t)$ 函数已经给出，这里在求 Laplace 变换前可以对 $u(t)$ 求各阶导数，则可以用下面的语句求出在给定的 $u(t)$ 下微分方程等号右侧的式子为

```
>> syms t; u=exp(-5*t)*cos(2*t+1)+5;   % 定义输入函数
   uu=laplace(5*diff(u,2)+4*diff(u)+2*u);   % 对等号右侧整个式子进行变换
   uu=collect(simple(uu)); latex(uu)   % 化简
```

用 LaTeX 可以更好地显示出等号右侧的式子为

$$\frac{290+(87\cos 1+92\sin 1+10)\,s^2+(619\cos 1+100+286\sin 1)\,s}{(s^2+10s+29)\,s}$$

得出了等号右面的式子，则可以等效地得出关于 $Y(s)$ 的有理函数式子，对之部分分式展开、求 Laplace 反变换并化简：

```
>> syms s; G=uu/(s^4+10*s^3+35*s^2+50*s+24);
   Y=residue(G,s); y=ilaplace(Y); latex(simple(y))
```

则可以最终得出如下的微分方程解析解为

$$y(t)=\left(\frac{97}{60}\sin 1+\frac{133}{30}\cos 1-\frac{5}{3}\right)\mathrm{e}^{-t}+\left(\frac{343}{520}\sin 1-\frac{547}{520}\cos 1\right)\mathrm{e}^{-5t}\sin(2t)+\frac{5}{12}+$$

$$\left(-\frac{547}{520}\sin 1-\frac{343}{520}\cos 1\right)\cos(2t)\,\mathrm{e}^{-5t}+\left(\frac{41}{15}\sin 1-\frac{271}{30}\cos 1+\frac{5}{12}\right)\mathrm{e}^{-4t}+$$
$$\left(-\frac{51}{13}\sin 1-\frac{445}{26}\cos 1+\frac{5}{2}\right)\mathrm{e}^{-2t}+\left(\frac{5}{8}\sin 1+\frac{179}{8}\cos 1-\frac{5}{3}\right)\mathrm{e}^{-3t}$$

若不采取直接给出部分分式展开及 Laplace 反变换的命令，用前面介绍的微分方程求解函数 dsolve() 也能得出完全等效的结果。具体的命令为

```
>> syms t; u=exp(-5*t)*cos(2*t+1)+5;     % 定义输入函数
   uu=laplace(5*diff(u,2)+4*diff(u)+2*u)    % 求出并显示等号右侧式子
uu =
   87*exp(-5*t)*cos(2*t+1)+92*exp(-5*t)*sin(2*t+1)+10
>> y1=dsolve(['D4y+10*D3y+35*D2y+50*Dy+24*y=',...
             '87*exp(-5*t)*cos(2*t+1)+92*exp(-5*t)*sin(2*t+1)+10'],...
            'y(0)=0','Dy(0)=0','D2y(0)=0','D3y(0)=0');
   simple(y-y1)   % 验证两种方法的结果完全一致
ans =
      0
```

有了微分方程的解析解，则可以用 ezplot() 函数直接绘制出该解的曲线，如图 7-1 所示。可见，ezplot() 函数可以直接处理得出的解。类似地，读者还可以绘制出输入信号曲线。注意，这里的初始条件限制了输出信号在 $t=0$ 时刻的变化。

```
>> ezplot(y,[0,10]);  axis([0,10,0,0.6])  % 手工调整坐标系范围
```

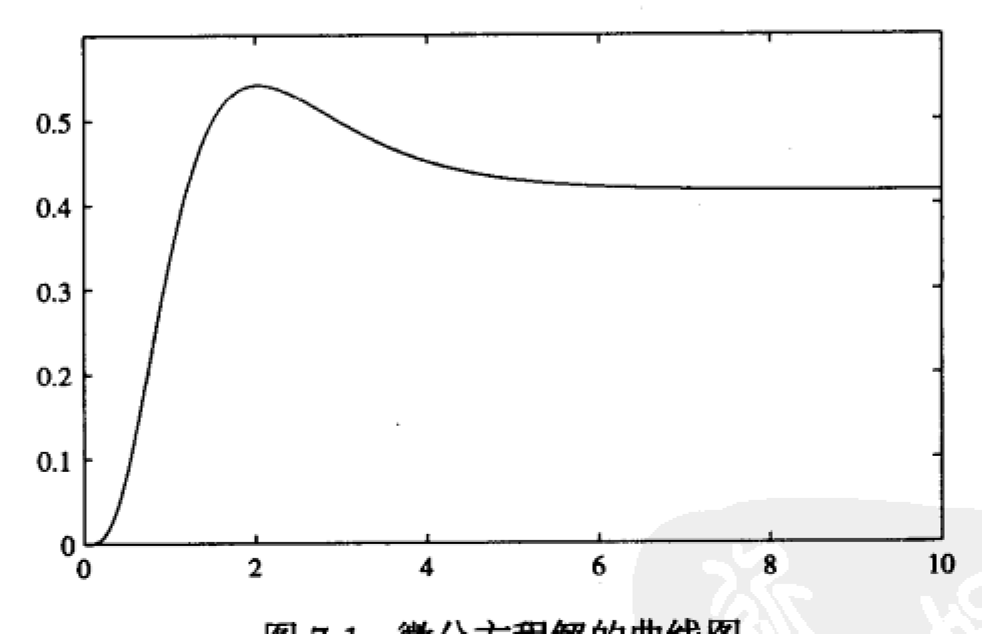


图 7-1 微分方程解的曲线图

7.1.4 特殊非线性微分方程的解析解

部分非线性微分方程也是可以用 dsolve() 函数求解析解的，这样的方程描述方式和前面介绍的线性微分方程是一致的，描述了这样的微分方程，则可以直接求解出微分方程的解析解。下面将通过例子演示非线性方程的解析解求解，同时还将演示不能求解的例子。

【例 7-5】 试求出一阶非线性微分方程 $\dot{x}(t)=x(t)(1-x^2(t))$ 的解析解。

【求解】 这样简单的一阶非线性方程可以考虑用 dsolve() 函数直接解出。

```
>> syms t x
   x=dsolve('Dx=x*(1-x^2)')
x =
    [  1/(1+exp(-2*t)*C1)^(1/2)]
    [ -1/(1+exp(-2*t)*C1)^(1/2)]
```

即该微分方程的解析解为 $x(t)=\pm 1/\sqrt{1+C_1\mathrm{e}^{-2t}}$。

其实，稍微改变原微分方程，例如将等号右侧加上 1，则可以用下面的语句试解该方程。读者会发现原始的微分方程是没有解析解的。

```
>> syms t x;  x=dsolve('Dx=x*(1-x^2)+1')
Warning: Explicit solution could not be found; implicit solution returned.
> In dsolve at 310
x =
   t+Int(-1/(a-a^3+1),a =  .. x)+C1 = 0
```

【例 7-6】 考虑著名的 Van der Pol 方程

$$\frac{\mathrm{d}^2y(t)}{\mathrm{d}t^2}+\mu(y^2(t)-1)\frac{\mathrm{d}y(t)}{\mathrm{d}t}+y(t)=0 \tag{7-1-6}$$

试用 dsolve() 函数求解它，看看能得出什么结论。

【求解】 由前面的讨论可见，似乎所有的微分方程都可以直接用 MATLAB 语言提供的强大的 dsolve() 函数求解，这样很自然地想到一般非线性微分方程的解析解问题。

对前面给出的 Van der Pol 方程，用户尝试如下的 MATLAB 命令，也将得出原微分方程无解析解的提示。

```
>> syms y mu; y=dsolve('D2y+mu*(y^2-1)*Dy+y')
y =
   &where(a,{[diff(b(a),a)*b(a)+mu*b(a)*a^2-mu*b(a)+a = 0, a = y(t),
   b(a) = diff(y(t),t), y(t) = a, t = Int(1/b(a),a)+C1]})
```

可见，微分方程解析解求解函数 dsolve() 并不能直接应用于一般非线性方程的解析解的求解。所以非线性微分方程只能用数值解法去求解，即使看起来很简单的非线性微分方程也是没有解析解的，只有极特殊的非线性微分方程解析可解。下面的内容将集中介绍各类非线性微分方程的数值解方法。

7.2 微分方程问题的数值解法

前面介绍了微分方程的解析解方法，同时也指出很多非线性微分方程是不存在解析解法的，需要使用数值解法对之进行研究。从本节开始着重讨论基于 MATLAB/Simulink 语言的各类微分方程的数值解方法。

7.2.1 微分方程问题算法概述

一般微分方程的数值解法很大一类是关于微分方程初值问题的数值解法，这类问题需要用一阶显式的微分方程组来描述为

$$\dot{\boldsymbol{x}}(t)=\boldsymbol{f}(t,\boldsymbol{x}(t)) \tag{7-2-1}$$

其中，$\boldsymbol{x}^{\mathrm{T}}(t)=[x_1(t),x_2(t),\cdots,x_n(t)]$ 称为状态向量，$\boldsymbol{f}^{\mathrm{T}}(\cdot)=[f_1(\cdot),f_2(\cdot),\cdots,f_n(\cdot)]$ 可以是任意非线性函数。所谓初值问题是指，若已知初始状态 $\boldsymbol{x}_0=[x_1(0),\cdots,x_n(0)]^{\mathrm{T}}$，用数值求解方法求出在某个时间区间 $t\in[0,t_f]$ 内各个时刻状态变量 $\boldsymbol{x}(t)$ 的数值，这里 t_f 又称为终止时间。

7.2.1.1 微分方程求解的误差与步长问题

对多元非线性常微分方程初值问题来说，Euler 算法是最直观的一类求解算法。虽然该算法比较简单，但理解该算法对理解其他复杂的微分方程算法是很有帮助的，故这里将以 Euler 算法为例介绍微分方程初值问题的数值算法。

假设已知在 t_0 时刻系统状态向量的值为 $\boldsymbol{x}(t_0)$，若选择一个很小的计算步长 h，则可以将微分方程左侧的导数近似为 $(\boldsymbol{x}(t_0+h)-\boldsymbol{x}(t_0))/(t_0+h-t_0)$，代入微分方程则可以解出在 t_0+h 时刻方程的近似解为

$$\hat{\boldsymbol{x}}(t_0+h)=\boldsymbol{x}(t_0)+h\boldsymbol{f}(t_0,\boldsymbol{x}(t_0)) \tag{7-2-2}$$

更严格地，因为这样的近似解是存在误差的，所以可以写出在 t_0+h 时刻系统状态向量的值为

$$\boldsymbol{x}(t_0+h)=\hat{\boldsymbol{x}}(t_0+h)+\boldsymbol{R}_0=\boldsymbol{x}_0+h\boldsymbol{f}(t,\boldsymbol{x}_0)+\boldsymbol{R}_0 \tag{7-2-3}$$

简记 $\boldsymbol{x}_1=\boldsymbol{x}(t_0+h)$，则 $\hat{\boldsymbol{x}}_1=\hat{\boldsymbol{x}}(t_0+h)$ 为系统状态向量在 t_0+h 时刻的近似值，亦即数值解。可见，$\boldsymbol{R}_0$ 为数值解的舍入误差。在实际解法中为简单起见，经常可以舍弃 ^ 记号，而将数值解直接记为 $\boldsymbol{x}_1$。

一般地，假设已知在 t_k 时刻系统的状态向量为 $\boldsymbol{x}_k$，则在 t_k+h 时刻 Euler 算法的数值解可以写成

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+h\boldsymbol{f}(t,\boldsymbol{x}_k) \tag{7-2-4}$$

这样，用迭代的方法可以由给定的初值问题逐步求出在所选择的时间段 $t\in[0,T]$ 内各个时刻 $t_0+h,t_0+2h,\cdots$ 处的原问题数值解。

提高数值解精度的一种方法是减小步长 h 的值。然而，并不能无限制地减小 h 的值，这主要有两个原因：

① **减慢计算速度** 因为对选定的求解时间而言，减小步长就意味着增加在这个时间段内的计算点数目，故计算速度减慢。

② **增加累积误差** 因为不论选择多小的步长，所得出的数值解都将有一个舍入误差，减小计算步长则将增加计算的次数，从而使得整个计算过程的舍入误差的叠加和传递次

数增多，产生较大的累积误差。舍入误差、累积误差和总误差关系的示意图如图 7-2 (a) 所示。

所以在对微分方程求解过程中，应采取下列措施：

① **选择适当的步长** 采用像 Euler 法这样简单的算法时，应适当地选择步长，既不能太大，又不能太小。

② **改进近似算法精度** 由于 Euler 算法只是将原积分问题进行梯形的近似，其近似精度很低，从而不能很有效地逼近原始问题。可以用各种更精确的插值方法来取代 Euler 算法，从而改进运算精度。比较成功的是 Runge-Kutta 法、Adams 法等。

③ **采用变步长方法** 前面提及“适当”地选择步长，这本身就是个模糊的概念，如何适当地选择步长取决于经验。事实上，很多种方法都允许变步长的求解，如果误差较小时，可自动地增大步长，而误差较大时再自动减小步长，从而精确、有效地求解给出的常微分方程初值问题。

一般变步长算法的原理如图 7-2 (b) 所示。已知 t_k 时刻的状态变量 $\boldsymbol{x}_k$，则先在某步长 h 下计算出 t_k+h 时刻的状态变量 $\tilde{\boldsymbol{x}}_{k+1}$。另外，将步长变成原来步长的一半，分两步从 $\boldsymbol{x}_k$ 计算出 t_k+h 时刻的状态变量 $\hat{\boldsymbol{x}}_{k+1}$。如果两种运算步长下的误差 $\epsilon=|\hat{\boldsymbol{x}}_{k+1}-\tilde{\boldsymbol{x}}_{k+1}|$ 小于给定的误差限，则可以采用该步长或适当增大步长，如果误差大，则进一步减小步长。自适应变步长算法可以较好地解决计算速度问题，另外能保证计算的精度。

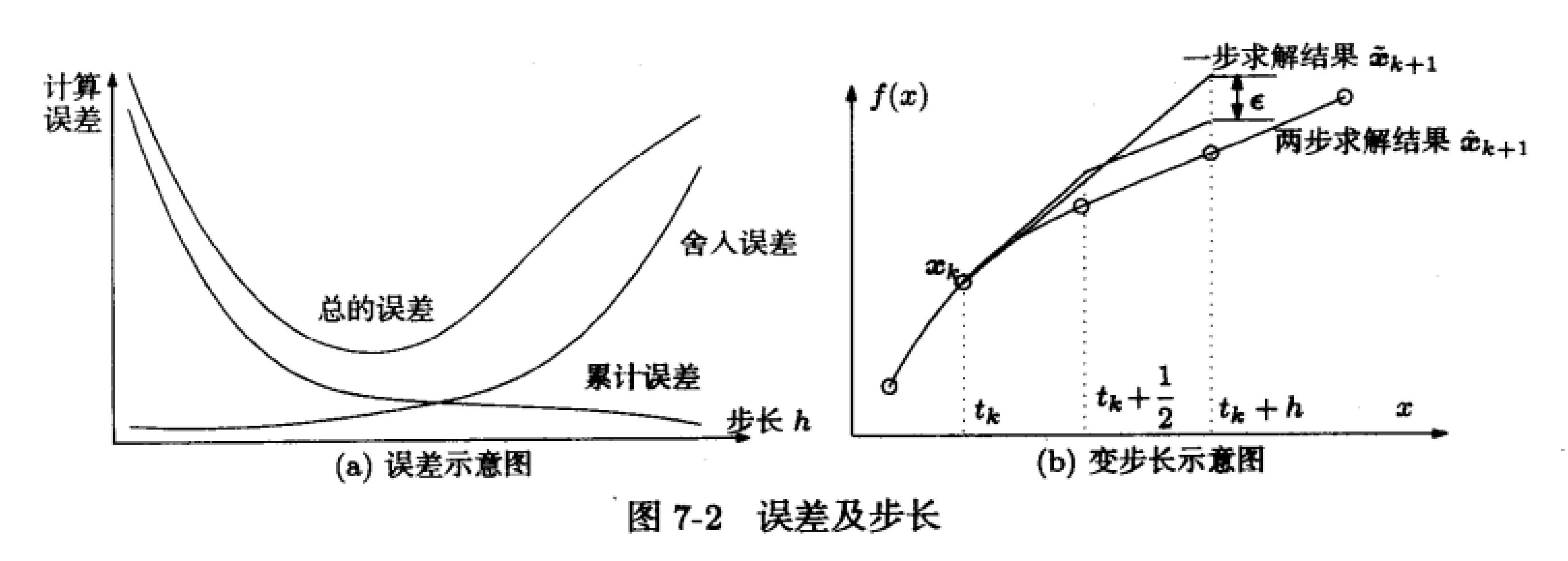


图 7-2 误差及步长

7.2.2 四阶定步长 Runge-Kutta 算法及 MATLAB 实现

四阶定步长的 Runge-Kutta 算法是传统数值分析课程和系统仿真课程中经常介绍的算法，被认为是求解微分方程的一种有效的方法，该算法结构很简单，可以先定义如下 4 个附加向量：

$$\begin{cases} \boldsymbol{k}_1 = h\boldsymbol{f}(t_k,\boldsymbol{x}_k) \\ \boldsymbol{k}_2 = h\boldsymbol{f}\left(t_k+\dfrac{h}{2},\boldsymbol{x}_k+\dfrac{\boldsymbol{k}_1}{2}\right) \\ \boldsymbol{k}_3 = h\boldsymbol{f}\left(t_k+\dfrac{h}{2},\boldsymbol{x}_k+\dfrac{\boldsymbol{k}_2}{2}\right) \\ \boldsymbol{k}_4 = h\boldsymbol{f}(t_k+h,\boldsymbol{x}_k+\boldsymbol{k}_3) \end{cases} \tag{7-2-5}$$

其中，h 为计算步长，在实际应用中该步长是一个常数，这样由四阶 Runge-Kutta 算法可以求解出下一个步长的状态变量值为

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\frac{1}{6}(\boldsymbol{k}_1+2\boldsymbol{k}_2+2\boldsymbol{k}_3+\boldsymbol{k}_4) \tag{7-2-6}$$

这样，用迭代的方法由给定的初值问题逐步求出在所选择的时间段 $t\in[t_0,t_h]$ 内各个时刻 $t_0+h,t_0+2h,\cdots$ 处的原问题数值解。

有了上面的数学算法，则可以用 MATLAB 语言容易地编写出该算法的函数如下：

```
function [tout,yout]=rk_4(odefile,tspan,y0)
t0=tspan(1); th=tspan(2);
if length(tspan)<=3, h=tspan(3);
else, h=tspan(2)-tspan(1); th=tspan(2); end
tout=[t0:h:th]'; yout=[];
for t=tout'
   k1=h*eval([odefile '(t,y0)']);
   k2=h*eval([odefile '(t+h/2,y0+0.5*k1)']);
   k3=h*eval([odefile '(t+h/2,y0+0.5*k2)']);
   k4=h*eval([odefile '(t+h,y0+k3)']);
   y0=y0+(k1+2*k2+2*k3+k4)/6;
   yout=[yout; y0'];
end
```

其中，tspan 可以有两种构成方法，第一种方法可以是一个等间距的时间向量；第二种方法是 tspan=[t_0，t_h，h]，其中，t_0 和 t_h 为计算的初始和终止值，h 为计算步长，odefile 是一个字符串变量，为表示微分方程的文件名，$\boldsymbol{y}_0$ 是初值列向量。函数调用完成后，时间向量与各个时刻状态变量构成的矩阵分别由 tout 和 yout 返回。

该算法看似简单，然而从求解数值问题的效果看，该算法不是一个较好的方法，故一般不使用该方法，后面将通过例子演示微分方程求解方法，并和现成的 MATLAB 函数进行对比分析。

7.2.3 一阶微分方程组的数值解

7.2.3.1 四阶五级 Runge-Kutta-Felhberg 算法

德国学者 Felhberg 对传统的 Runge-Kutta 方法进行了改进[8]，在每一个计算步长内对 $f_i(\cdot)$ 函数进行 6 次求值，以保证更高的精度和数值稳定性，该算法又称为四阶五级 FRK 算法。假设当前的步长为 h_k，则可以定义下面的 6 个 $\boldsymbol{k}_i$ 变量：

$$\boldsymbol{k}_i=h_k\boldsymbol{f}\left(t_k+\alpha_ih_k,\boldsymbol{x}_k+\sum_{j=1}^{i-1}\beta_{ij}\boldsymbol{k}_j\right),\ i=1,2,\cdots,6 \tag{7-2-7}$$

式中，t_k 为当前计算时刻，中间参数 α_i, β_{ij} 及其他参数由表 7-1 给出，其中，α_i, β_{ij} 参数对又称为 Dormand-Prince 对。这时下一步的状态向量可以由下式求出。

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \sum_{i=1}^{6} \gamma_i \boldsymbol{k}_i \tag{7-2-8}$$

表 7-1 四阶五级 RKF 算法系数表

α_i	β_{ij}					γ_i	γ_i^*
0						16/135	25/216
1/4	1/4					0	0
3/8	3/32	9/32				6656/12825	1408/2565
12/13	1932/2197	-7200/2197	7296/2197			28561/56430	2197/4104
1	439/216	-8	3680/513	-845/4104		-9/50	-1/5
1/2	-8/27	2	-3544/2565	1859/4104	-11/40	2/55	0

当然，直接采用这一方法为定步长的方法。在实际问题中往往希望在一些情况下 (如解变化很快时) 采用较小的步长，而在另一些情况下 (如解的变化很缓慢时) 采用较大的步长，这样做既可以保证较高的精度，又可以保证较高的运算速度。在此算法中，定义一个误差向量 $\boldsymbol{\epsilon}_k = \sum_{i=1}^{6}(\gamma_i - \gamma_i^*)\boldsymbol{k}_i$，可以由它的大小来变换计算步长，所以这种能自动变换步长的方法又称为自适应变步长方法。

定步长算法相当于用开环控制的思想求解微分方程，选定步长后将不考虑计算是不是有误差，也不考虑误差是否能被接受，一味采用单一步长计算全程。而变步长算法则是采用由误差作为监测指标的闭环控制思想，在计算过程中修正步长，使得预期的计算精度得以保证，同时由于采用定步长思想，所以大部分问题的求解时间将明显缩短，因为在计算过程中在保证计算精度的前提下也允许大步长。

7.2.3.2 基于 MATLAB 的微分方程求解函数

MATLAB 下求解一阶微分方程组初值问题数值解的最常用的方法是 `ode45()` 函数，该函数实现了前面介绍的变步长四阶五级 Runge-Kutta-Felhberg 算法，可以采用变步长的算法求解微分方程。该函数的调用格式为

```
[t, x]=ode45(Fun,[t0,tf],x0)                      直接求解
[t, x]=ode45(Fun,[t0,tf],x0,options)              带有控制选项
[t, x]=ode45(Fun,[t0,tf],x0,options,p1,p2,…)      带有附加参数
```

其中，微分方程应该用 MATLAB 函数 `Fun` 或 `inline()` 函数按指定的格式描述，有关该函数的编写方法后面将通过例子专门介绍，$[t_0, t_f]$ 描述微分方程求解的区间，如果只给出一个值 t_f 则表示初始时刻为 $t_0 = 0$，终止时刻为 t_f 的问题求解。为使得微分方程能够求解，还应该已知初值问题的初始状态变量 $\boldsymbol{x}_0$。另外，该函数还允许 $t_f < t_0$，即可以认为

t_0 为终值时刻，t_f 为初始时刻，故 $\boldsymbol{x}_0$ 表示状态变量的终值，故该函数可以直接求解终值问题。

求解一阶显式微分方程组的关键是用 MATLAB 语言编写一个函数，描述需要求解的微分方程组。该函数的入口应该为

```
function x_d=funname(t,x)                        不需附加变量的格式
function x_d=funname(t,x,flag,p_1,p_2,···)        可以使用附加变量
```

其中，t 是时间变量或自变量，即使需要求解的微分方程不是时变的，也需要给出 t 占位，否则 MATLAB 在变量传递过程中将出现问题。变量 $\boldsymbol{x}$ 为状态向量，返回的 $\boldsymbol{x}_d$ 为状态向量的导数。flag 一般用来控制求解过程，可以用其给初始状态赋值。

在微分方程求解中有时需要对求解算法及控制条件进行进一步设置，这可以通过求解过程中的 options 变量进行修改。初始 options 变量可以通过 odeset() 函数获取，该变量是一个结构体变量，其中有众多成员变量，表 7-2 中列出了常用的一些成员变量。

表 7-2 常微分方程求解函数的控制参数表

参数名	参 数 说 明
RelTol	为相对误差容许上限，默认值为 0.001 (即 0.1% 的相对误差)，在一些特殊的微分方程求解中，为了保证较高的精度，还应该再适当减小该值
AbsTol	为一个向量，其分量表示每个状态变量允许的绝对误差，其默认值为 10^{-6}。当然可以自由设置其值，以改变求解精度
MaxStep	为求解方程最大允许的步长
Mass	微分代数方程中的质量矩阵，可用于描述微分代数方程
Jacobian	为描述 Jacobi 矩阵函数 $\partial \boldsymbol{f}/\partial \boldsymbol{x}$ 的函数名，如果已知该 Jacobi 矩阵，则能加速仿真过程

修改该变量有两种方式，其一是用 odeset() 函数设置，其二是直接修改 options 的成员变量。例如，若想将相对误差设置成较小的 10^{-7}，则需要给出

```
options=odeset('RelTol',1e-7);              或更直观地
options=odeset; options.RelTol=1e-7;
```

在实际求解过程中经常需要定义一些附加参数，这些参数由 $p_1,p_2,\cdots,p_m$ 表示，在编写方程函数时也应该一一对应地写出，在这种调用格式下还应该使用 flag 变量占位。后面将通过例子详细介绍相关的调用格式。

【例 7-7】假设著名的 Lorenz 模型的状态方程表示为

$$\begin{cases}\dot{x}_1(t)=-\beta x_1(t)+x_2(t)x_3(t)\\ \dot{x}_2(t)=-\rho x_2(t)+\rho x_3(t)\\ \dot{x}_3(t)=-x_1(t)x_2(t)+\sigma x_2(t)-x_3(t)\end{cases}$$

其中，设 $\beta=8/3,\rho=10,\sigma=28$。若令其初值为 $x_1(0)=x_2(0)=0,\ x_3(0)=\epsilon$，而 ϵ 为机器上可以识别的小常数，如取一个很小的正数 $\epsilon=10^{-10}$，试求解该微分方程组。

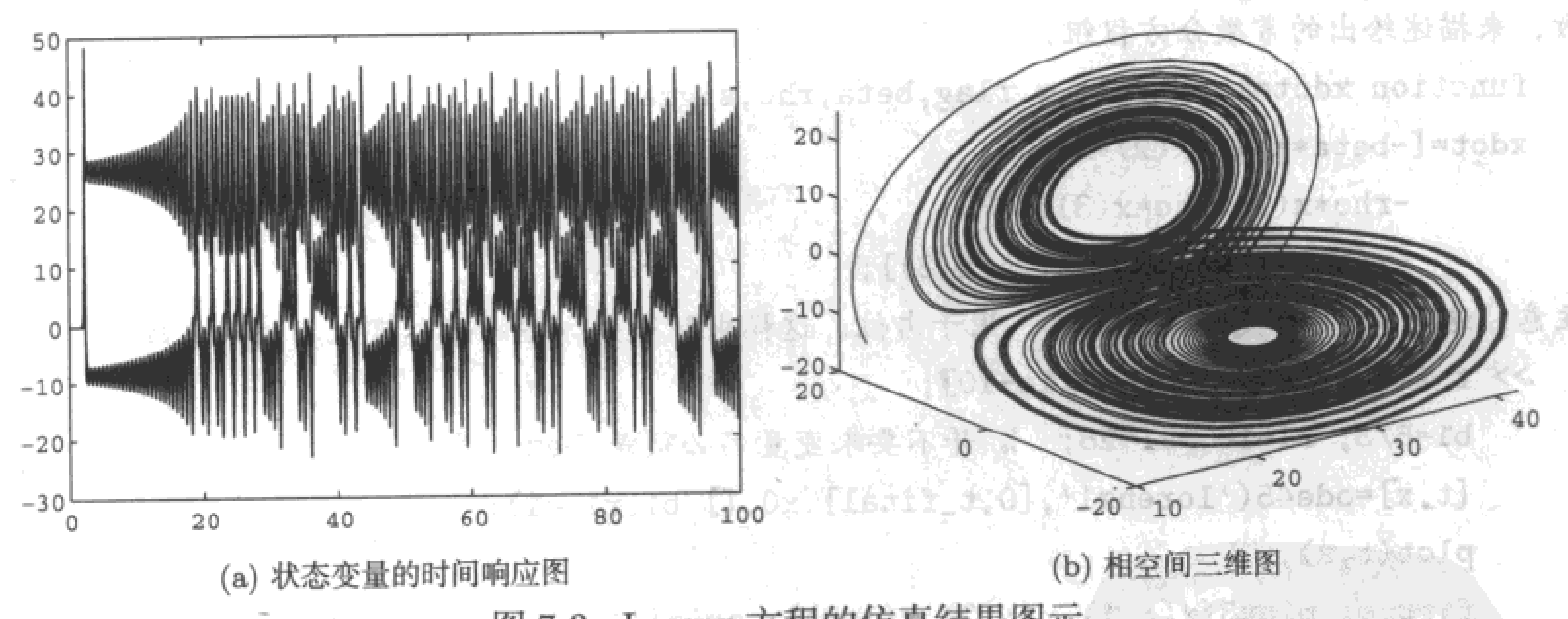
【求解】 该方程是非线性微分方程，所以不存在解析解，只能用数值解法求解。若想用数值算法求解该微分方程，可以按下面的格式编写出一个 MATLAB 函数 lorenzeq.m 来描述系统的动态模型。其内容为

```
function xdot = lorenzeq(t,x)
xdot=[-8/3*x(1)+x(2)*x(3);
      -10*x(2)+10*x(3);
      -x(1)*x(2)+28*x(2)-x(3)];
```

这时，可以调用微分方程数值解 ode45() 函数对 lorenzeq() 函数描述的系统进行数值求解，并将结果进行图形显示。

```
>> t_final=100; x0=[0;0;1e-10];
   [t,x]=ode45('lorenzeq',[0,t_final],x0);plot(t,x),
   figure;  % 打开新图形窗口
   plot3(x(:,1),x(:,2),x(:,3));
   axis([10 42 -20 20 -20 25]);  % 根据实际数值手动设置坐标系
```

其中，t_final 为设定的仿真终止时间，x0 为初始状态。第一个绘图命令绘制出系统的各个状态和时间关系的二维曲线图，如图 7-3 (a) 所示。第二个绘图命令可以绘制出三个状态的相空间曲线，如图 7-3 (b) 所示。可以看出，看似很复杂的三元一阶常微分方程组的数值解问题由几条简单直观的 MATLAB 语句就求解出来了。此外，用 MATLAB 语言还可以轻易、直观地将结果用可视化方式直接显示出来，这就是选择 MATLAB 语言作为本书主要语言的原因。



(a) 状态变量的时间响应图　　(b) 相空间三维图

图 7-3　Lorenz 方程的仿真结果图示

其实，观察相空间轨迹走行的最好方法是采用 comet3() 函数绘制动画式的轨迹，故可将最后一个语句改成 comet3(x(:,1),x(:,2),x(:,3))。

从前面的微分方程求解实例看，如果能建立一个描述微分方程组的 MATLAB 函数或 inline() 函数，则调用 ode45() 可以立即解出方程的解析解。可以看出，编写一个 MATLAB 函数来描述微分方程是常微分方程初值问题数值求解的关键。

【例 7-8】 考虑例 7-7 中给出的 Lorenz 方程，试用 inline() 函数描述该方程，并求出该微分方程的数值解，与前面的结果进行比较。

【求解】 可以用 lorenzeq() 函数或用下面的语句直接定义该方程。

```
>> f1=inline(['[-8/3*x(1)+x(2)*x(3); -10*x(2)+10*x(3);',...
     '-x(1)*x(2)+28*x(2)-x(3)]'],'t','x');
```

其中，f1 为生成的函数名，可以用 ode45() 这类微分方程求解函数直接调用。这里自变量仍为 t 和 x，微分方程的函数内容一行显示不下，故将其折成两行显示。定义了微分方程模型，就可以通过下面的语句求解这个微分方程了。采用下面的命令，将得出与例 7-7 完全一致的结果。

```
>> t_final=100; x0=[0;0;1e-10]; [t,x]=ode45(f1,[0,t_final],x0);
   plot(t,x), figure;
   plot3(x(:,1),x(:,2),x(:,3)); axis([10 42 -20 20 -20 25]);
```

7.2.3.3 MATLAB 下带有附加参数的微分方程求解

在基于 MATLAB 语言的微分方程求解中，引入附加参数的目的是，若微分方程的某些参数可以选择不同的值，对不同值求解时考虑附加参数可以避免每次修改模型文件。例如，例 7-7 中给出的 Lorenz 方程中，β,ρ,σ 可以看成附加参数，这样在表示它们的变化时，无需修改描述原始微分方程的 MATLAB 函数，而只需在调用微分方程求解时从外部修改它们的参数即可。

【例 7-9】 试编写带有附加参数的 MATLAB 函数来描述例 7-7 中的 Lorenz 方程，并利用该函数研究分别求解在该例中给定参数下和一组新参数 $\beta=2,\rho=5,\sigma=20$ 下 Lorenz 方程的数值解。

【求解】 选定附加参数为 β,ρ,σ，根据上述的微分方程描述格式，可以编写出如下的 MATLAB 函数，来描述给出的常微分方程组。

```
function xdot=lorenz1(t,x,flag,beta,rho,sigma)
xdot=[-beta*x(1)+x(2)*x(3);
      -rho*x(2)+rho*x(3);
      -x(1)*x(2)+sigma*x(2)-x(3)];
```

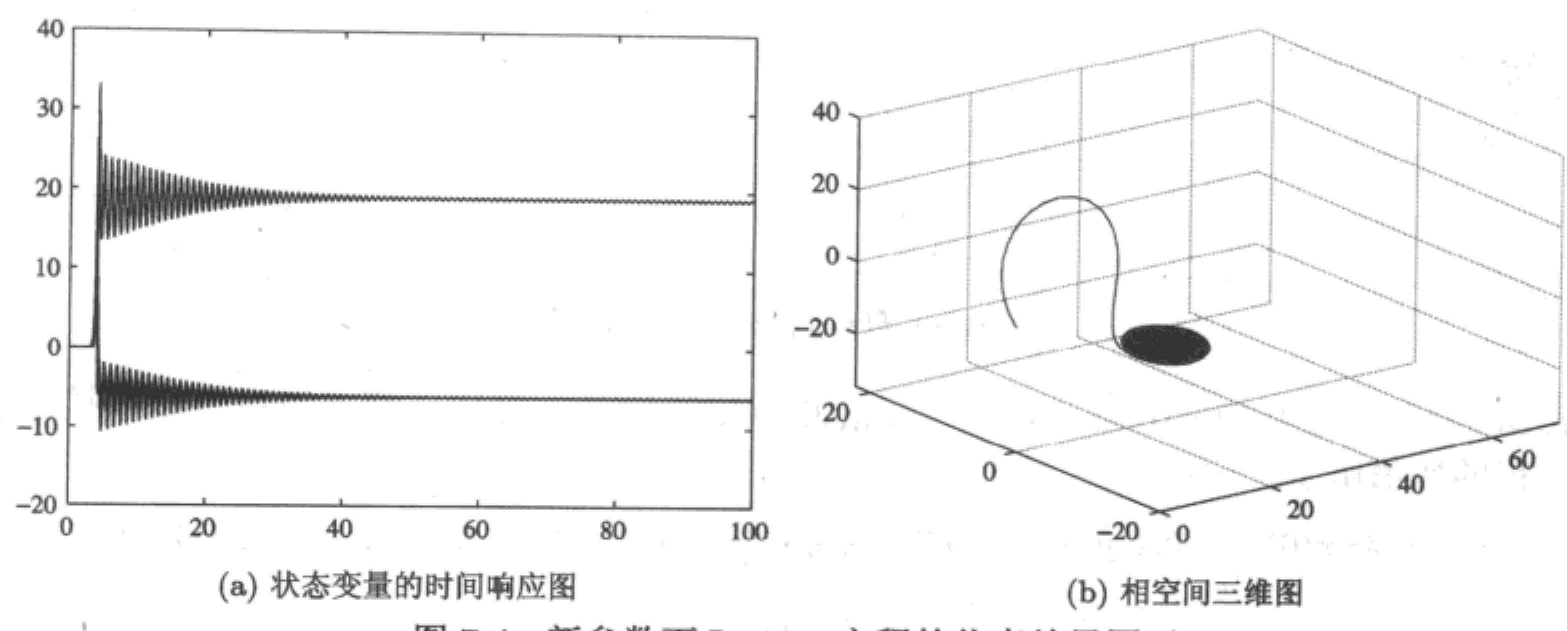
注意，这里的 flag 变量不能省略，用于占位。这样求解微分方程的语句可以类似地修改为

```
>> t_final=100; x0=[0;0;1e-10];
   b1=8/3; r1=10; s1=28;  % 并不要求变量名必须是 beta 等
   [t,x]=ode45('lorenz1',[0,t_final],x0,[],b1,r1,s1);
   plot(t,x),
   figure; plot3(x(:,1),x(:,2),x(:,3)); axis([10 42 -20 20 -20 25]);
```

从上面的调用格式可以看出，调用函数时无需使用和函数本身完全一致的变量名，只要对应关系正确就可以了。另外，在 options 的位置上给出了空矩阵 []，表示不需要修改控制选项。

有了带有附加参数的微分方程 M 函数后，就可以在其他的参数值 β,ρ,σ 下求解微分方程，而无需改变 lorenz1.m 文件。例如，若选择 $\beta=1,\rho=5,\sigma=20$，则可以用下面的语句直接求出数值解，这样将分别得出如图 7-4 (a)、图 7-4 (b) 所示的二维和三维图形。

```
>> t_final=100; x0=[0;0;1e-10];
   b2=2; r2=5; s2=20;
   [t2,x2]=ode45('lorenz1',[0,t_final],x0,[],b2,r2,s2);
```



(a) 状态变量的时间响应图　　(b) 相空间三维图

图 7-4　新参数下 Lorenz 方程的仿真结果图示

```
plot(t2,x2),
figure; plot3(x2(:,1),x2(:,2),x2(:,3)); axis([0 72 -20 22 -35 40]);
```

inline() 函数同样也可以支持带有附加变量的微分方程描述方式。例如，前面用 MATLAB 函数形式表示的 lorenz1() 文件可以用 inline() 函数描述成

```
>> f2=inline(['[-beta*x(1)+x(2)*x(3); -rho*x(2)+rho*x(3);',...
     '-x(1)*x(2)+sigma*x(2)-x(3)]'],'t','x','flag','beta','rho','sigma');
```

注意，flag 变量是不能省略的。这时，相应的求解语句为

```
>> t_final=100; x0=[0;0;1e-10];
   b2=2; r2=5; s2=20;
   [t2,x2]=ode45(f2,[0,t_final],x0,[],b2,r2,s2);
   plot(t2,x2),
   figure; plot3(x2(:,1),x2(:,2),x2(:,3)); axis([0 72 -20 22 -35 40]);
```

而得出的结果与图 7-4 (a)、图 7-4 (b) 完全一致。可见，这样处理的好处是不必再编写一个 MATLAB 函数来描述 Lorenz 微分方程了，直接用 MATLAB 命令就可以得出结果。

7.2.4　微分方程转换

由前面介绍的微分方程求解函数和微分方程标准型可见，如果常微分方程由一个或多个高阶常微分方程给出，要得出该方程的数值解，则应该先将该方程变换成一阶常微分方程组。这里将分两种情况加以考虑。

7.2.4.1　单个高阶常微分方程处理方法

假设一个高阶常微分方程的一般形式为

$$y^{(n)} = f(t, y, \dot{y}, \cdots, y^{(n-1)}) \tag{7-2-9}$$

且已知输出变量 $y(t)$ 的各阶导数初始值为 $y(0), \dot{y}(0), \cdots, y^{(n-1)}(0)$，则可以选择一组状态变量 $x_1 = y, x_2 = \dot{y}, \cdots, x_n = y^{(n-1)}$，这样，就可以将原高阶常微分方程模型变换成下面

的一阶方程组形式

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \quad \vdots \\ \dot{x}_n = f(t, x_1, x_2, \cdots, x_n) \end{cases} \tag{7-2-10}$$

且初值 $x_1(0) = y(0), x_2(0) = \dot{y}(0), \cdots, x_n(0) = y^{(n-1)}(0)$。这样，变换以后可以直接求取原方程的数值解了。

【例 7-10】 已知 $y(0) = -0.2, \dot{y}(0) = -0.7$，试用数值方法求出的 Van der Pol 方程的解。

$$\ddot{y} + \mu(y^2 - 1)\dot{y} + y = 0$$

【求解】 例 7-6 中已经演示过，该方程没有解析解，所以不能用解析方法求解该方程，只有依靠数值解法了。因为该方程不是由 MATLAB 直接可解的一阶显式微分方程组给出的，所以需要对该方程进行变换。选择状态变量 $x_1 = y, x_2 = \dot{y}$，则原方程可以变换成

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\mu(x_1^2 - 1)x_2 - x_1 \end{cases}$$

这里的 μ 是一个可变参数，如果对每一个要研究的 μ 值都编写一个函数则显得不方便，所以应该采用附加参数的概念将 μ 的值传给该函数，这样可以如下写出描述此模型的 M 函数为

```
function y=vdp_eq(t,x,flag,mu)
y=[x(2);
   -mu*(x(1)^2-1)*x(2)-x(1)];
```

可见，在函数定义时多了一个 mu 项，该项应该在 ode45() 函数调用时传给 vdp_eq() 函数。假定初值为 $\boldsymbol{x} = [-0.2, -0.7]^{\mathrm{T}}$，则最终的求解函数格式为

```
>> x0=[-0.2; -0.7]; t_final=20;
   mu=1; [t1,y1]=ode45('vdp_eq',[0,t_final],x0,[],mu);
   mu=2; [t2,y2]=ode45('vdp_eq',[0,t_final],x0,[],mu);
   plot(t1,y1,t2,y2,':')
   figure; plot(y1(:,1),y1(:,2),y2(:,1),y2(:,2),':')
```

这样，在 $\mu = 1, 2$ 时的时间响应曲线和相平面曲线分别如图 7-5 (a)、图 7-5 (b) 所示。

注意，在定义函数时应该有个 flag 变量，其作用是用来指定初值的。即使初值不用指定，也必须有该变量占位。调用函数 ode45() 时也应该给出选项变量占位。在 ode45() 调用命令中的附加变量个数应该和方程 M 函数中的附加参数个数完全对应，否则将出现错误结果。

改变 μ 的值，令 $\mu = 1000$，并设仿真终止时间为 3000，则可以采用下面的命令求解相应的 Van der Pol 方程。

```
>> x0=[2;0]; t_final=3000;
   mu=1000; [t,y]=ode45('vdp_eq',[0,t_final],x0,[],mu);
```

经过长时间的等待，发现得出下面的错误信息：

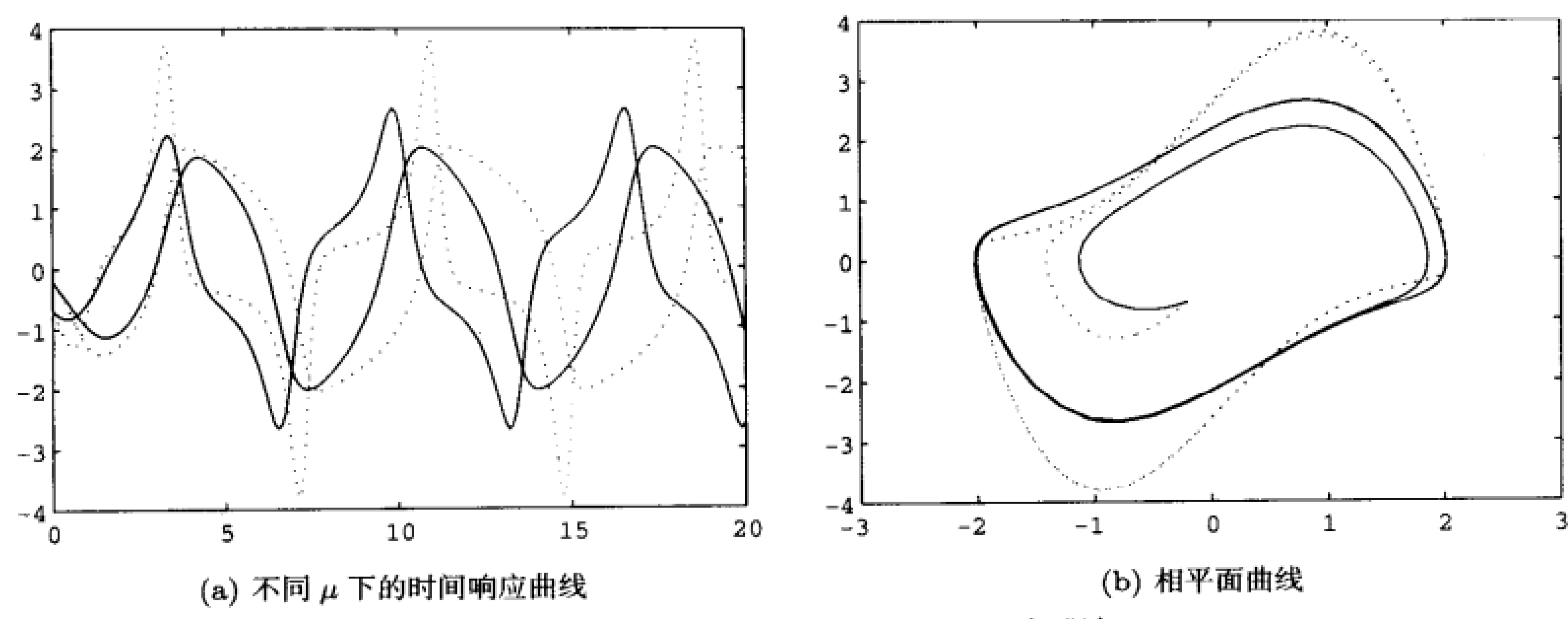


图 7-5 不同 μ 值下 Van der Pol 方程解

```
??? Error using ==> vertcat
Out of memory. Type HELP MEMORY for your options.
Error in ==> c:\matlab7\toolbox\matlab\funfun\ode45.m
On line 392  ==>           yout = [yout; zeros(chunk,neq)];
```

事实上，由于变步长所采用的步长过小，而要求的仿真终止时间比较大，导致输出的 $\boldsymbol{y}$ 矩阵过大，超出了计算机存储空间的容限。所以，这个问题不适合采用 ode45() 来求解，后面将采用刚性方程求解的算法来解决这个问题。

7.2.4.2 高阶常微分方程组的变换方法

这里以两个高阶微分方程构成的微分方程组为例介绍如何将之变换成一个一阶显式常微分方程组。如果可以显式地将两个方程写成

$$\begin{cases} x^{(m)} = f(t, x, \dot{x}, \cdots, x^{(m-1)}, y, \cdots, y^{(n-1)}) \\ y^{(n)} = g(t, x, \dot{x}, \cdots, x^{(m-1)}, y, \cdots, y^{(n-1)}) \end{cases} \tag{7-2-11}$$

则仍旧可以选择状态变量 $x_1 = x, x_2 = \dot{x}, \cdots, x_m = x^{(m-1)}, x_{m+1} = y, x_{m+2} = \dot{y}, \cdots, x_{m+n} = y^{(n-1)}$，这样就可以将原方程变换成

$$\begin{cases} \dot{x}_1 = x_2 \\ \quad\vdots \\ \dot{x}_m = f(t, x_1, x_2, \cdots, x_{m+n}) \\ \dot{x}_{m+1} = x_{m+2} \\ \quad\vdots \\ \dot{x}_{m+n} = g(t, x_1, x_2, \cdots, x_{m+n}) \end{cases} \tag{7-2-12}$$

再对初值进行相应的变换，就可以得出所期望的一阶微分方程组了。下面将通过一个例子演示常微分方程组的转换与求解。

【例 7-11】已知 Apollo 卫星的运动轨迹 (x, y) 满足下面的方程

$$\ddot{x}=2\dot{y}+x-\frac{\mu^*(x+\mu)}{r_1^3}-\frac{\mu(x-\mu^*)}{r_2^3},\ \ddot{y}=-2\dot{x}+y-\frac{\mu^* y}{r_1^3}-\frac{\mu y}{r_2^3}$$

其中，$\mu=1/82.45$, $\mu^*=1-\mu$, $r_1=\sqrt{(x+\mu)^2+y^2}$, $r_2=\sqrt{(x-\mu^*)^2+y^2}$，试在初值 $x(0)=1.2$, $\dot{x}(0)=0$, $y(0)=0$, $\dot{y}(0)=-1.04935751$ 下进行求解，并绘制出 Apollo 位置的 (x, y) 轨迹。

【求解】选择一组状态变量 $x_1=x, x_2=\dot{x}, x_3=y, x_4=\dot{y}$，这样就可以得出一阶常微分方程组为

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=2x_4+x_1-\mu^*(x_1+\mu)/r_1^3-\mu(x_1-\mu^*)/r_2^3\\ \dot{x}_3=x_4\\ \dot{x}_4=-2x_2+x_3-\mu^* x_3/r_1^3-\mu x_3/r_2^3\end{cases}$$

式中，$r_1=\sqrt{(x_1+\mu)^2+x_3^2}, r_2=\sqrt{(x_1-\mu^*)^2+x_3^2}$，且 $\mu=1/82.45, \mu^*=1-\mu$。

有了数学模型描述，则可以立即写出其相应的 MATLAB 函数如下：

```
function dx=apolloeq(t,x)
mu=1/82.45; mu1=1-mu;
r1=sqrt((x(1)+mu)^2+x(3)^2); r2=sqrt((x(1)-mu1)^2+x(3)^2);
dx=[x(2);
     2*x(4)+x(1)-mu1*(x(1)+mu)/r1^3-mu*(x(1)-mu1)/r2^3;
     x(4);
    -2*x(2)+x(3)-mu1*x(3)/r1^3-mu*x(3)/r2^3];
```

调用 ode45() 函数可以求出该方程的数值解为

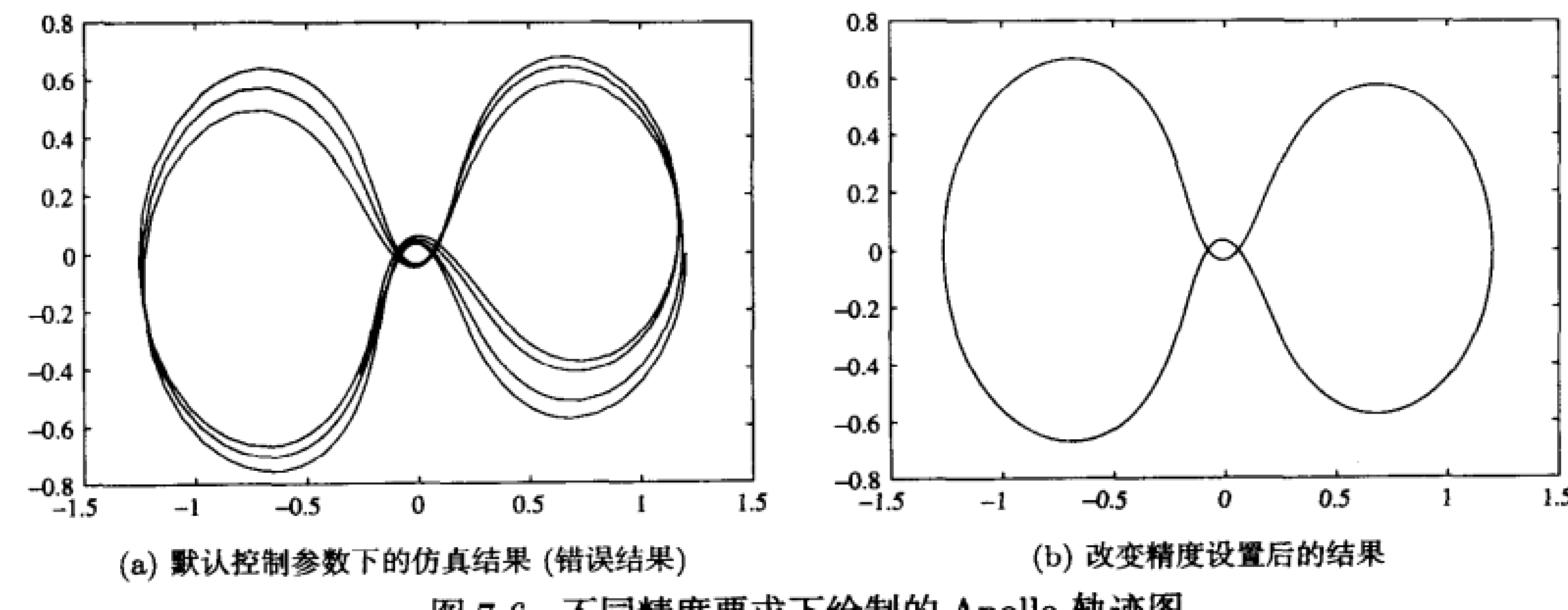
```
>> x0=[1.2; 0; 0; -1.04935751];
   tic, [t,y]=ode45('apolloeq',[0,20],x0); toc
   length(t), plot(y(:,1),y(:,3))
Elapsed time is 0.330000 seconds.
ans =
        689
```

得出的轨迹如图 7-6 (a) 所示。

其实，这样直接得出的 Apollo 轨道是不正确的，因为这时 ode45() 函数选择的默认精度控制 RelTol 设置得太大，从而导致较高的误差传递。可以减小该值，直至减小到 0.000001，使用下面的语句进行仿真研究。

```
>> options=odeset; options.RelTol=1e-6;
   tic, [t1,y1]=ode45('apolloeq',[0,20],x0,options); toc
   length(t1), plot(y1(:,1),y1(:,3)),
Elapsed time is 0.701000 seconds.
ans =
        1873
```

将得出的轨迹如图 7-6 (b) 所示。可见，在新的默认精度下结果是完全不同的。这时，再进一步减小精度控制误差限也不会有太大的改进了。所以在仿真结束后有时有必要减小精度误差限 RelTol，看看得出的结果是否还相同，依此判定这样选择的精度要求是否合适，否则对求解结果是否正确没有把握。



(a) 默认控制参数下的仿真结果 (错误结果)　　(b) 改变精度设置后的结果

图 7-6　不同精度要求下绘制的 Apollo 轨迹图

用下面的 MATLAB 命令还求出求解全程所采用的最小步长，并可以绘制出计算步长的曲线，如图 7-7 所示。

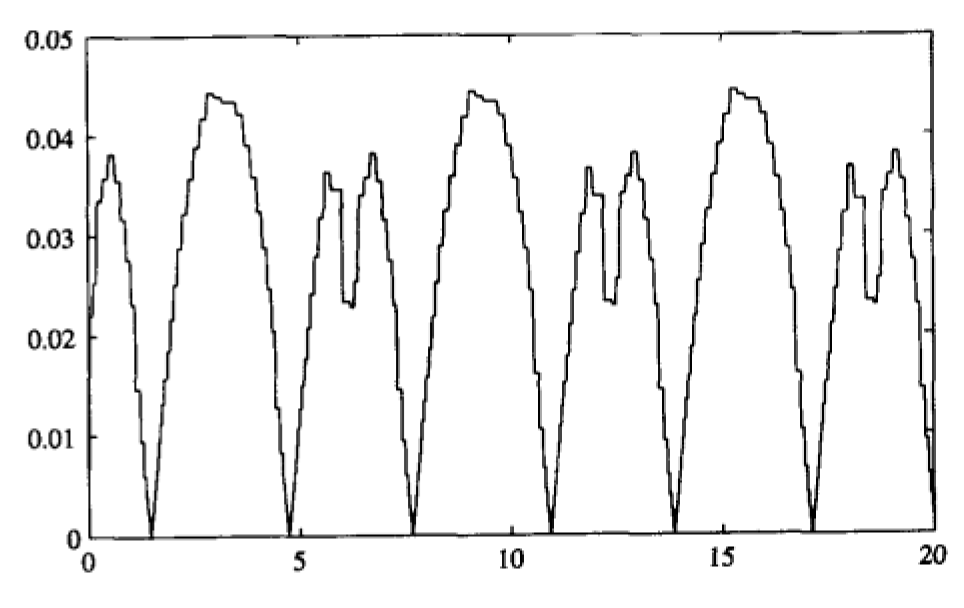


图 7-7　仿真过程中的计算步长

```
>> min(diff(t1))
ans =
    1.892665624865231e-004
>> plot(t1(1:end-1),diff(t1))
```

从得出的图形可以看出变步长算法的意义，即在需要小步长时取小步长，而变换缓慢时取大步长计算，这样就可以保证以高效率求解方程。

由给出的计算步长图可以看出，部分时间段用了大于 0.04 的步长，这时一般定步长计算问题求解中不敢用的大步长。为了保证某些具体时间点上的计算精度，还会自动采用 2×10^{-4} 的小步长。换言之，在这些点上如果采用比该值大的步长，则计算误差就不能保证在 10^{-6} 之下。考虑定

步长计算的方式，如果想保证 10^{-6} 之下的误差，全程选择的步长就应该是这样的值，这样计算的点就要达到 10^5 个，是这里变步长算法的 56 倍。

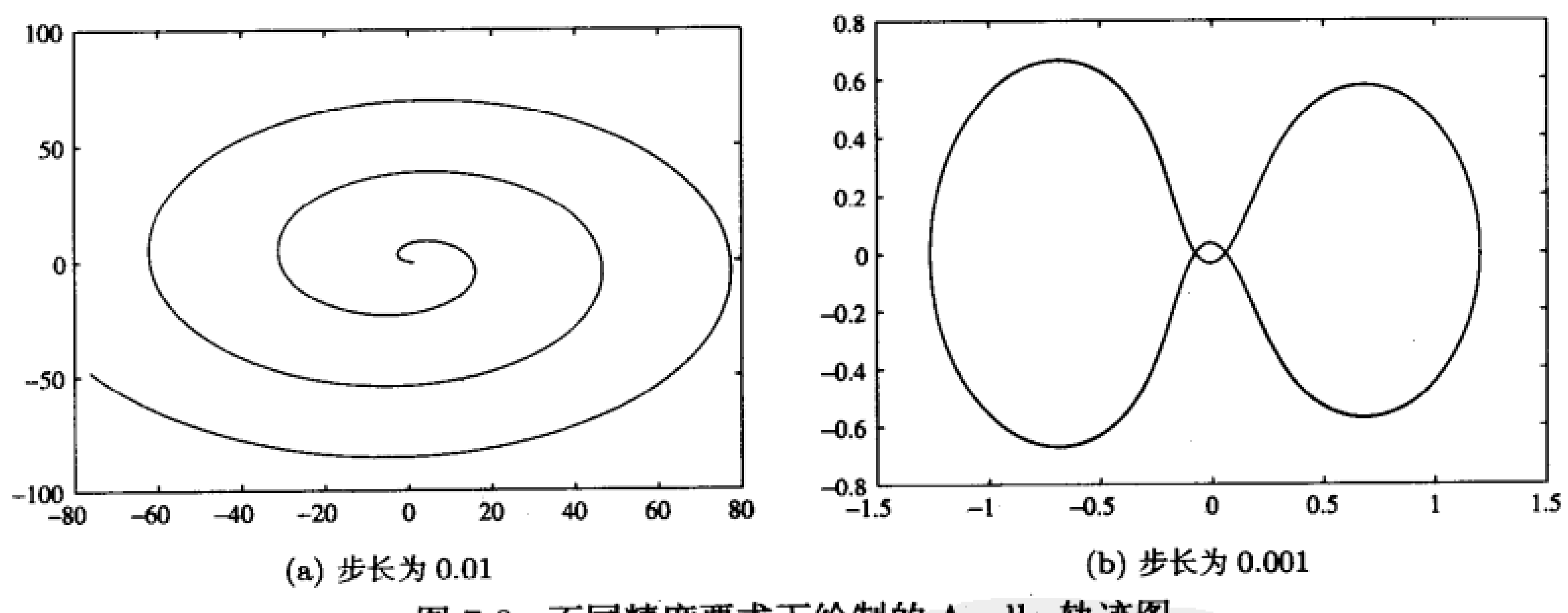
【例 7-12】试用定步长的四阶 Runge-Kutta 算法求解 Apollo 微分方程方程。

【求解】用定步长的方法求解微分方程将面临两个问题：①如何选择步长；②如何确保求解的精度。前一个问题可以通过试凑的方法，从步长选择的角度看，选择很小的步长对保证求解精度有利，但计算量会明显增加。对前面介绍的 Apollo 轨迹方程，可以试着选择步长为 0.01，则用下面语句可以求解微分方程，并绘制出 Apollo 轨迹曲线，如图 7-8 (a) 所示。

```
>> x0=[1.2; 0; 0; -1.04935751];
   tic, [t1,y1]=rk_4('apolloeq',[0,20,0.01],x0); toc
   plot(y1(:,1),y1(:,3))      % 绘制出轨迹曲线
Elapsed time is 2.654000 seconds.
```

显而易见，这样求解的结果是错误的，应该采用更小的步长求解，直至选择步长为 0.001，则可以求解微分方程，并绘制出更精确的轨迹曲线，如图 7-8 (b) 所示，但求解时间达到 97 秒，是变步长算法的 138 倍。

```
>> tic, [t2,y2]=rk_4('apolloeq',[0,20,0.001],x0); toc
   plot(y2(:,1),y2(:,3))      % 绘制出轨迹曲线
Elapsed time is 97.079000 seconds.
```



(a) 步长为 0.01　　(b) 步长为 0.001

图 7-8 不同精度要求下绘制的 Apollo 轨迹图

其实，上面的结果在某些点上严格说来仍不能满足 10^{-6} 的误差限，只是形似变步长得出的结果，所以在求解常微分方程组时建议采用变步长算法，而没有必要自己按照数值分析类课程中介绍的定步长算法去编写程序求解。

如果两个高阶微分方程都同时隐式地含有 $x^{(m)}$ 和 $y^{(n)}$ 项，则首先需要对之进行相应的处理，然后再用上述的方法进行最终变换。下面将通过一个例子加以说明。

【例 7-13】假设系统模型以二元方程组形式给出

$$\begin{cases} \ddot{x}+2\dot{y}x=2\ddot{y} \\ \ddot{x}\dot{y}+3\dot{x}\ddot{y}+x\dot{y}-y=5 \end{cases}$$

试将其转换成一阶微分方程组。

【求解】 可见，这两个方程均同时含有 $\ddot{x}$ 和 $\ddot{y}$，所以仍可以选择一组状态变量 $x_1=x, x_2=\dot{x}, x_3=y, x_4=\dot{y}$，我们的目的是先消去其中一个高阶导数，求解第一个式子，得出 $\ddot{y}$ 的解析表达式为

$$\ddot{y}=\dot{y}x+\frac{\ddot{x}}{2}$$

然后将它代入第二个式子中，则可以解出 $\ddot{x}$ 为

$$\ddot{x}=\frac{2y+10-2x\dot{y}-6x\dot{x}\dot{y}}{2\dot{y}+3\dot{x}}$$

这样可以写出其状态方程表示为

$$\dot{x}_2=\frac{2x_3+10-2x_1x_4-6x_1x_2x_4}{2x_4+3x_2}$$

将上面的结果再代入到 $\ddot{y}$ 的方程中，得

$$\dot{x}_4=\frac{x_3+5-x_1x_4+2x_1x_4^2}{2x_4+3x_2}$$

综上所述，可以列写出方程的一阶微分方程组表示为

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=\dfrac{2x_3+10-2x_1x_4-6x_1x_2x_4}{2x_4+3x_2}\\ \dot{x}_3=x_4\\ \dot{x}_4=\dfrac{x_3+5-x_1x_4+2x_1x_4^2}{2x_4+3x_2}\end{cases}$$

事实上，这样的方程还是不太容易手工求解的，但可以依赖 MATLAB 的符号运算工具箱来求解该问题。为了方便求解起见，记 dx=$\ddot{x}$, dy=$\ddot{y}$，这样，dx 和 dy 实际上还是 $\dot{x}_2$ 和 $\dot{x}_4$，故可以用下面的语句得出方程的解为

```
>> syms x1 x2 x3 x4
   [dx,dy]=solve('dx+2*x4*x1=2*dy','dx*x4+3*x2*dy+x1*x4-x3=5','dx,dy');
   latex(dx), latex(dy)
```

用 LaTeX 排版语言可以得出如下的显示结果：

$$\dot{x}_2=-2\frac{3x_4x_1x_2-5+x_4x_1-x_3}{3x_2+2x_4},\ \ \dot{x}_4=\frac{2x_4^2x_1+5-x_4x_1+x_3}{3x_2+2x_4}$$

可见，得出的结果与前面手工得出的完全一致。

对于更复杂的问题来说，手工变换的难度将很大，所以如果可能，可以采用计算机去求解有关方程，获得解析解。如果不能获得方程的解析解，也需要在描写一阶常微分方程组时列写出式子，得出问题的数值解。

7.3 特殊微分方程的数值解

从第 7.2 节的介绍和例子看，一般常微分方程组均可以转换成一阶显式常微分方程组，然后通过给定的算法及 MATLAB 求解函数，如 ode45() 直接求出方程的数值解。从前面的例子还可以看出，ode45() 函数有时失效，所以应该引入一类方程，即刚性微分方程的专门求解函数来解决这样的问题。另外，微分代数方程、隐式微分方程及延迟微分方程也是需要引入的微分方程类型，它们的求解将弥补 ode45() 函数本身的不足，本节将着重介绍这些方程的求解问题。

7.3.1 刚性微分方程的求解

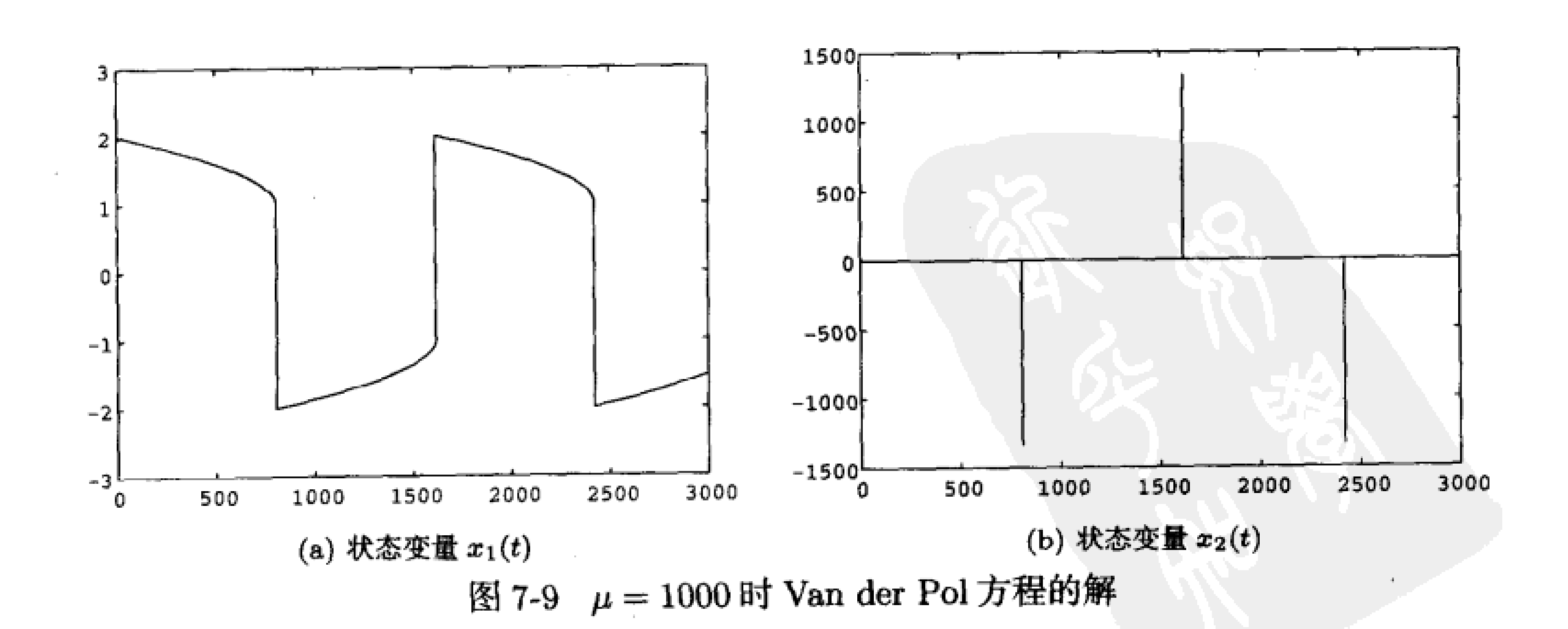
在许多领域中，经常遇到一类特殊的常微分方程，其中一些解变化缓慢，另一些变化快，且相差较悬殊，这类方程常常称为刚性方程，又称为 Stiff 方程。刚性问题一般不适合由 ode45() 这类函数求解，而应该采用 MATLAB 求解函数 ode15s()，该函数的调用格式和 ode45() 完全一致。

【例 7-14】 试求解 $\mu=1000$ 时的 Van der Pol 方程的数值解。

【求解】 仿照前面的例子可以给出如下的 MATLAB 命令:

```
>> h_opt=odeset; h_opt.RelTol=1e-6; x0=[2;0]; t_final=3000;
   tic, mu=1000; [t,y]=ode15s('vdp_eq',[0,t_final],x0,h_opt,mu); toc
Elapsed time is 1.943000 seconds.
>> plot(t,y(:,1)); figure; plot(t,y(:,2))
```

可见，用刚性方程求解函数可以快速求出该方程的数值解，并将两个状态变量的时间曲线分别绘制出来，如图 7-9 所示。从得出的图形可以看出，$x_1(t)$ 曲线变化较平滑，而 $x_2(t)$ 变化在某些点上较快，所以当 $\mu=1000$ 时，Van der Pol 方程属于典型的刚性方程，应该采用刚性方程的函数求解。



(a) 状态变量 $x_1(t)$　(b) 状态变量 $x_2(t)$

图 7-9 $\mu=1000$ 时 Van der Pol 方程的解

【例 7-15】 在传统的有关常微分方程数值解的教科书[49] 中，都认为下面的微分方程是刚性的。

$$\dot{\boldsymbol{y}}=\begin{bmatrix}-21 & 19 & -20\\ 19 & -21 & 20\\ 40 & -40 & -40\end{bmatrix}\boldsymbol{y},\ \boldsymbol{y}_0=\begin{bmatrix}1\\0\\-1\end{bmatrix}$$

试用 MATLAB 语言求解该微分方程。

【求解】 该方程的解析解可以通过 MATLAB 符号运算工具箱中语句直接求出。

```
>> syms t; A=sym([-21,19,-20; 19,-21,20; 40,-40,-40]);
   y0=[1; 0; -1]; y=expm(A*t)*y0
```

原方程的解析解为

$$\boldsymbol{y}(t)=\begin{bmatrix}0.5\mathrm{e}^{-2t}+0.5\mathrm{e}^{-40t}(\cos 40t+\sin 40t)\\ 0.5\mathrm{e}^{-2t}-0.5\mathrm{e}^{-40t}(\cos 40t+\sin 40t)\\ \mathrm{e}^{-40t}(\sin 40t-\cos 40t)\end{bmatrix}$$

现在考虑该问题的数值求解方法。根据原始问题，可以立即写出该模型的 MATLAB 表示为

```
function dy=c7xstf1(t,x)
dy=[-21,19,-20; 19,-21,20; 40,-40,-40]*x;
```

利用下面的 MATLAB 语句，可以得出方程的数值解为

```
>> opt=odeset; opt.RelTol=1e-6;
   tic,[t,y]=ode45('c7xstf1',[0,1],[1;0;-1],opt); toc
Elapsed time is 0.160000 seconds.
>> x1=exp(-2*t); x2=exp(-40*t).*cos(40*t); x3=exp(-40*t).*sin(40*t);
   y1=[0.5*x1+0.5*x2+0.5*x3, 0.5*x1-0.5*x2-0.5*x3, -x2+x3];
   plot(t,y,t,y1,':')
```

原方程的解析解和数值解如图 7-10 (a) 所示。可以看出，问题的数值解的精度还是比较高的，计算速度相对也较快，但从这里似乎看不出原问题的刚性所在。究其原因，因为在 MATLAB 下采用了变步长的算法，它可以依照要求的精度自动地修正步长，所以感受不到它是个刚性问题。

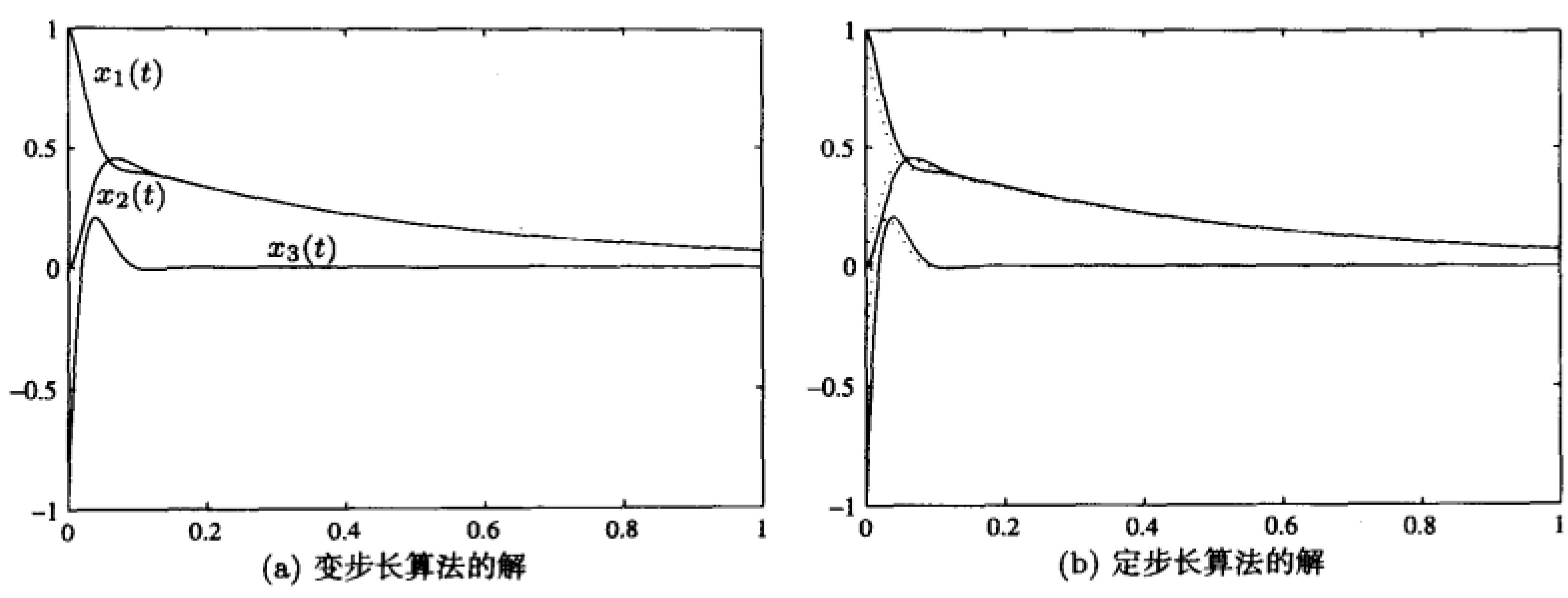
如果采用定步长方法，利用前面编写的四阶 Runge-Kutta 定步长算法程序 rk_4()，再用下面的语句就能求解原方程了。

```
>> tic,[t2,y2]=rk_4('c7xstf1',[0,1,0.01],[1;0;-1]); toc
   plot(t,y1,t2,y2,':')
Elapsed time is 0.210000 seconds.
```

这样得出的曲线与解析解的对比见图 7-10 (b)。从计算的结果看，显然采用定步长的算法在取较大的步长时，得出的解是不正确的。减小步长时，直至减小到 0.0001 时，仍能从得出的结果看出与解析解的差距，而这时的计算时间为 26 秒，是前面 ode45() 变步长算法所需时间的 162 倍。所以在实际应用最好采用变步长算法。

从得出的曲线可以看出，相比之下，这 3 条曲线的变化速度差别不是特别悬殊，在以前计算能力受限时被误认为是刚性问题，而在 MATLAB 下则可以由普通的函数求解。

由此可以得出结论，许多传统的刚性问题采用 MATLAB 的普通求解函数就可以直接



(a) 变步长算法的解 (b) 定步长算法的解

图 7-10 给定的传统刚性问题解法比较

解出，而不必刻意地去选择刚性问题的解法。当然，在有些问题的求解中确实需要采用刚性问题的解法，这将在例 7-16 中加以说明。

【例 7-16】考虑下面的常微分方程

$$\begin{cases} \dot{y}_1 = 0.04(1-y_1)-(1-y_2)y_1+0.0001(1-y_2)^2 \\ \dot{y}_2 = -10^4 y_1 + 3000(1-y_2)^2 \end{cases}$$

其中，该方程的初值为 $y_1(0)=0, y_2(0)=1$。取计算区间为 $t \in (0,100)$，试选择合适的算法得出该方程的数值解。

【求解】根据给出的微分方程，可以写出下面的 MATLAB 函数：

```
function dy=c7exstf2(t,y)
dy=[0.04*(1-y(1))-(1-y(2))*y(1)+0.0001*(1-y(2))^2;
    -10^4*y(1)+3000*(1-y(2))^2];
```

在 MATLAB 的命令窗口中给出下面的语句：

```
>> tic,[t2,y2]=ode45('c7exstf2',[0,100],[0;1]); toc
Elapsed time is 312.119000 seconds.
>> length(t2), plot(t2,y2)
ans =
    356941
```

经过长时间的等待，可以得出原问题的数值解，如图 7-11 (a) 所示。可以看出，调用普通的解法函数 ode45() 计算所需的时间过长，计算的点也过多，对这个例子来说，计算的点高达 35 万。再分析变步长解法所使用的步长，语句如下：

```
>> format long, [min(diff(t2)), max(diff(t2))]
ans =
   0.00022220693884   0.00214971787184
>> plot(t2(1:end-1),diff(t2))
```

则可以看出，由于设定的精度要求较高，不得不采用小步长来解决问题。实际的步长如图 7-11

(b) 所示。可见在大部分时间内，所采用的步长小于 0.0004，这使得解题时间大大增加。

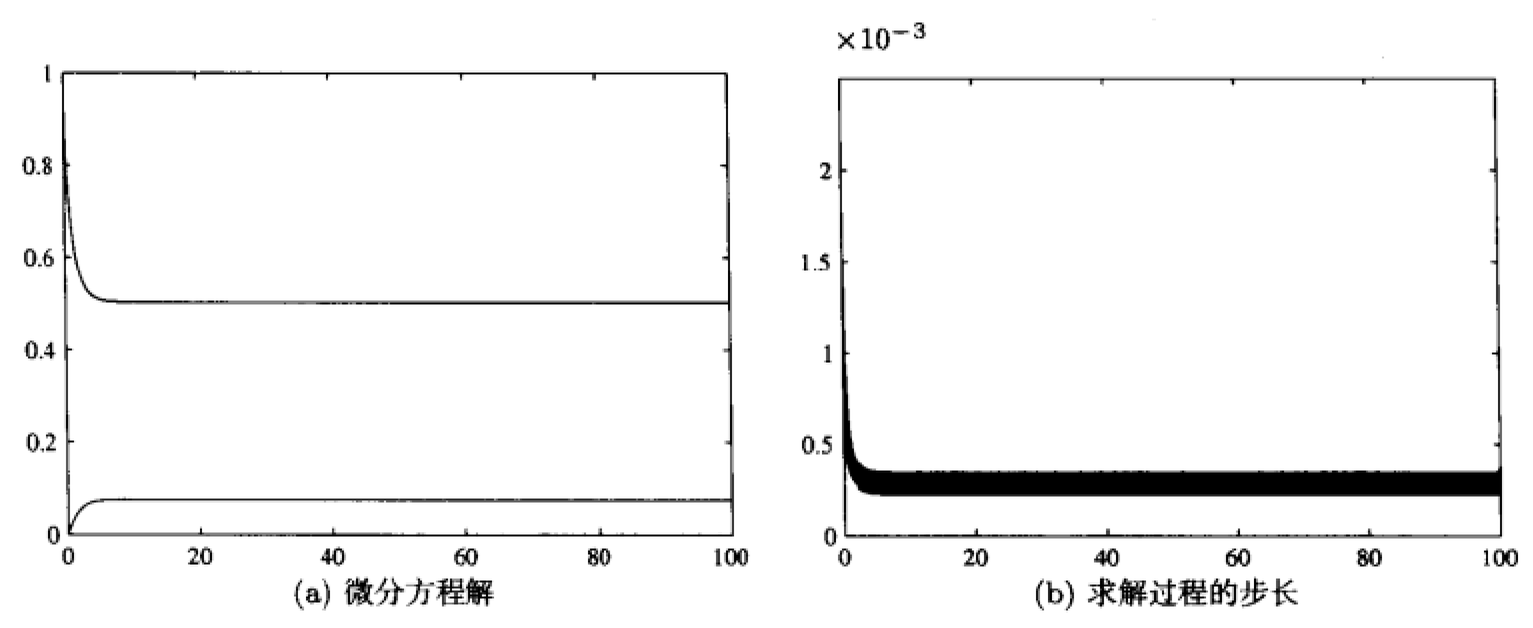


图 7-11 四阶五级 Runge-Kutta-Felhberg 法结果

考虑用 ode15s() 替代 ode45()，可以得出如下的结果：

```
>> opt=odeset; opt.RelTol=1e-6;
   tic,[t1,y1]=ode15s('c7exstf2',[0,100],[0;1],opt); toc
Elapsed time is 0.281000 seconds.
>> length(t1), plot(t1,y1)
ans =
   169
```

可见，解题时间大大减小，效率提高了 1100 多倍，得出的曲线则几乎完全一致。

7.3.2 隐式微分方程求解

所谓隐式微分方程就是那些不能转换成式 (7-2-1) 中一阶显式常微分方程组的微分方程。MATLAB 语言早期版本中未提供能直接求解隐式微分方程的算法及函数，所以仍可以借用显式微分方程求解的函数来求解这类问题。本节将通过两个例子来演示隐式微分方程的求解方法与应用。

【例 7-17】给定的隐式微分方程为

$$\begin{cases} \sin x_1\dot{x}_1 + \cos x_2\dot{x}_2 + x_1 = 1 \\ -\cos x_2\dot{x}_1 + \sin x_1\dot{x}_2 + x_2 = 0 \end{cases}$$

已知 $x_1(0) = x_2(0) = 0$，试求出该方程的数值解。

【求解】令 $\boldsymbol{x} = [x_1, x_2]^{\mathrm{T}}$，则可以将原方程改写成矩阵形式 $\boldsymbol{A}(\boldsymbol{x})\dot{\boldsymbol{x}} = \boldsymbol{B}(\boldsymbol{x})$，其中，

$$\boldsymbol{A}(\boldsymbol{x}) = \begin{bmatrix} \sin x_1 & \cos x_2 \\ -\cos x_2 & \sin x_1 \end{bmatrix},\ \boldsymbol{B}(\boldsymbol{x}) = \begin{bmatrix} 1 - x_1 \\ -x_2 \end{bmatrix}$$

如果能证明 $\boldsymbol{A}(\boldsymbol{x})$ 为非奇异的矩阵，则直接能将该方程变换成标准的显式一阶微分方程组的形式，即 $\dot{\boldsymbol{x}} = \boldsymbol{A}^{-1}(\boldsymbol{x})\boldsymbol{B}(\boldsymbol{x})$，套用各种 MATLAB 函数即可求解对应的方程。事实上，由于无法

严格证明 $\boldsymbol{A}(\boldsymbol{x})$ 矩阵是非奇异矩阵，故可以大胆地假设该矩阵为非奇异矩阵，利用 MATLAB 语言试解该方程，如果在求解过程中不出现 $\boldsymbol{A}(\boldsymbol{x})$ 为奇异矩阵的警告信息，则说明对求出的解不存在 $\boldsymbol{A}(\boldsymbol{x})$ 为奇异矩阵的现象，故得出的解是有效的。如果求解过程中确实出现矩阵奇异的信息，则得出的解没有实际意义。

对于这里要研究的隐式微分方程模型，可以编写下面的 MATLAB 函数：

```
function dx=c7ximp(t,x)
A=[sin(x(1)) cos(x(2)); -cos(x(2)) sin(x(1))];
B=[1-x(1); -x(2)]; dx=inv(A)*B;
```

这样，可以给出下面的命令求解方程：

```
>> opt=odeset; opt.RelTol=1e-6;
   [t,x]=ode45('c7ximp',[0,10],[0; 0],opt); plot(t,x)
```

同时将绘制出状态变量的时间曲线，如图 7-12 所示。在求解的过程中也没有得出有关矩阵奇异的错误信息，故得出的结果是可信的。

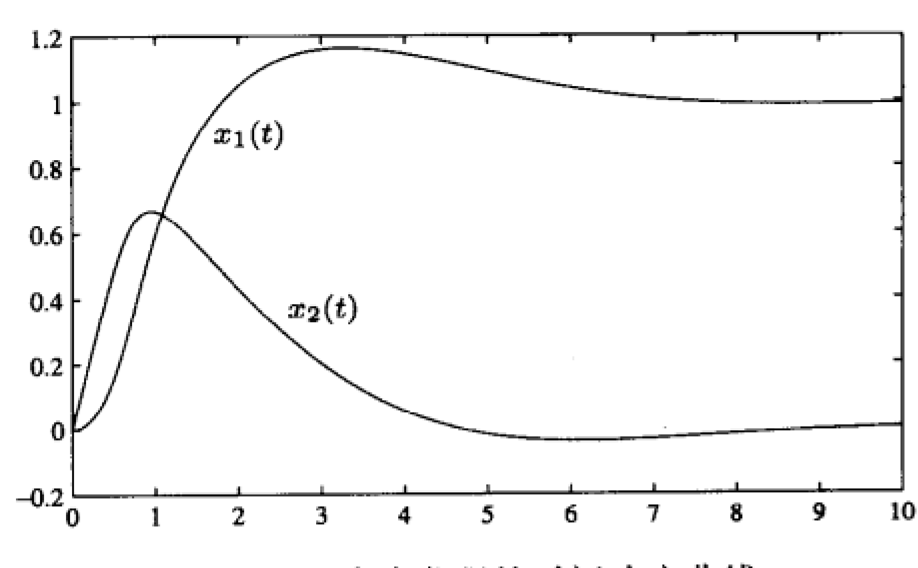


图 7-12 隐式方程的时间响应曲线

【例 7-18】 前面的例子很简单，可以将其立即变换成显式微分方程。现在考虑一个更复杂的隐式微分方程组

$$\begin{cases} \ddot{x}\sin\dot{y} + \ddot{y}^2 = -2xy + x\ddot{x}\dot{y} \\ x\ddot{x}\ddot{y} + \cos\ddot{y} = 3y\dot{x} \end{cases}$$

假设该方程具有初始状态为 $\boldsymbol{x} = [1,0,0,1]^{\mathrm{T}}$，试求取该微分方程的数值解。

【求解】 可以仍然选定状态变量 $x_1 = x, x_2 = \dot{x}, x_3 = y, x_4 = \dot{y}$，这样仍然有 $\dot{x}_1 = x_2, \dot{x}_3 = x_4$。显然，并不可能像例 7-13 那样解析地求解方程，得出 $\dot{x}_2$ 和 $\dot{x}_4$ 的显式表达式，但可以讨论该方程组数值解的方法对每组状态变量 $\boldsymbol{x}$ 得出它们的值。

由原来给出的方程，假设 $p_1 = \ddot{x}$, $p_2 = \ddot{y}$，则可以将原方程改写成

$$\begin{cases} p_1\sin x_4 + p_2^2 + 2x_1x_3 - x_1p_1x_4 = 0 \\ x_1p_1p_2 + \cos p_2 - 3x_3x_2 = 0 \end{cases}$$

依据该方程可以得出如下的 MATLAB 语句，从而写出相应的函数来描述微分方程，如下：

```
function dy=c7impode(t,x)
```

```
dx=inline(['[p(1)*sin(x(4))+p(2)^2+2*x(1)*x(3)-x(1)*p(1)*x(4);',...
        'x(1)*p(1)*p(2)+cos(p(2))-3*x(3)*x(2)]'],'p','x');
ff=optimset; dx1=fsolve(dx,x([1,3]),ff,x);
dy=[x(2); dx1(1); x(4); dx1(2)];
```

进入该函数时，由状态变量 $\boldsymbol{x}$ 和新定义的 p_1,p_2 可以写出 inline() 函数，描述未知量 p_i 代入方程后两个方程的误差，而这里 $\boldsymbol{x}$ 是作为已知的附加变量给出的。微分方程求解程序每次调用这个原型函数时均求解一次关于 p_i 的代数方程，得出的结果 p_1,p_2 实际上就是 $\dot{x}_2,\dot{x}_4$，这样就可以构造出微分方程对应的状态变量的导数。在求解代数方程中使用了一个小技巧，即代数方程的初值选择 $p_1(0)=x_1,p_2(0)=x_3$，这样会使得代数方程收敛速度和精度都加快。

建立起微分方程模型后，就可以通过下面的语句直接求解微分方程，并绘制出状态变量的时间曲线，如图 7-13 所示。

```
>> [t,x]=ode15s('c7impode',[0,2],[1,0,0,1]); plot(t,x)
```

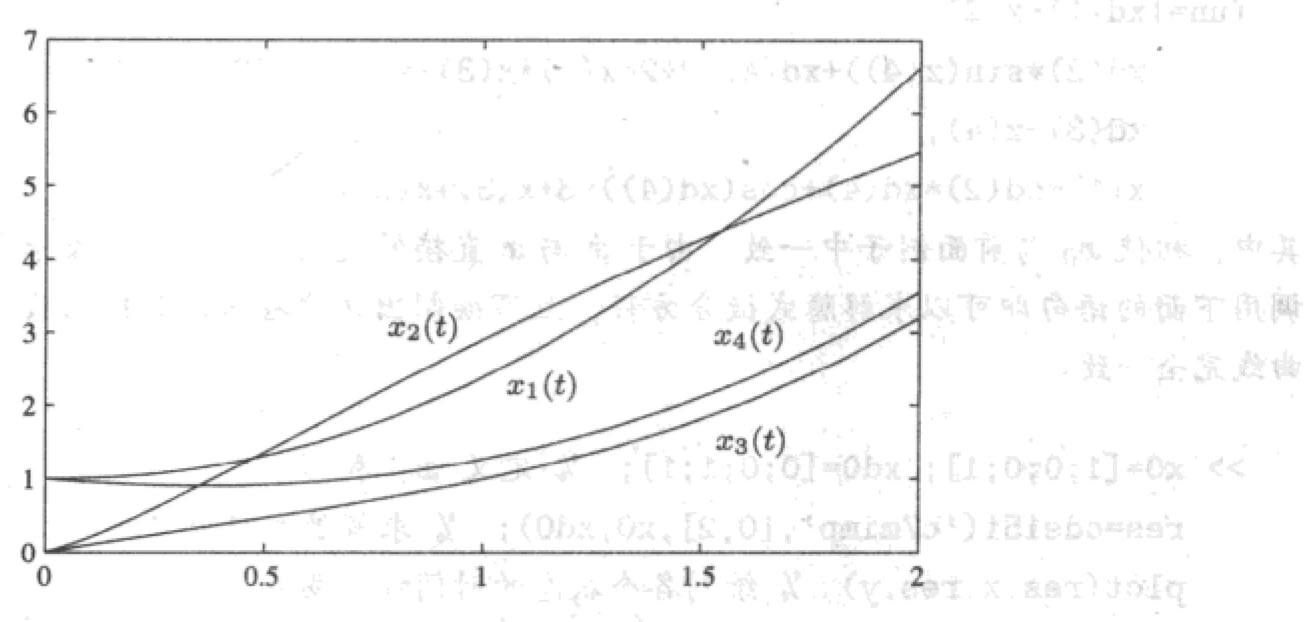


图 7-13　隐式方程的时间响应曲线

MATLAB 7.0 版本的新函数 ode15i() 可以直接用于隐式微分方程的求解。若隐式微分方程的数学描述为

$$\boldsymbol{F}[t,\boldsymbol{x}(t),\dot{\boldsymbol{x}}(t)]=\boldsymbol{0},\quad 且\ \boldsymbol{x}(0)=\boldsymbol{x}_0,\ \dot{\boldsymbol{x}}(0)=\dot{\boldsymbol{x}}_0 \tag{7-3-1}$$

则可以编写一个函数 fun() 描述该隐式微分方程，然后用 decic() 函数求出未完全定义的初值条件，再调用 ode15i() 函数即可以求解该隐式微分方程如下：

```
[x0*,ẋ0*]=decic(fun,t0,x0,x0^F,ẋ0,ẋ0^F)
res=ode15i(fun,tspan,x0*,ẋ0*)
```

隐式微分方程不同于一般显示微分方程，求解之前需要给出 $(\boldsymbol{x}_0,\dot{\boldsymbol{x}}_0)$，它们不能任意赋值，只能有 n 个是独立的，其余的需要用隐式方程求解，否则将可能出现矛盾的初始条件。所以在实际求解过程中，如果不能确定 $\dot{\boldsymbol{x}}_0^*$ 值，则应该先调用 decic() 函数得出相容的初值。在函数调用中 $(\boldsymbol{x}_0,\dot{\boldsymbol{x}}_0)$ 为任意给定的初值，$\boldsymbol{x}_0^F$ 和 $\dot{\boldsymbol{x}}_0^F$ 均为 n 维列向量，其值为 1 表示需要保留的初值，为 0 表示需要求解的初值项，通过方程求解将得出相容的初值

$\boldsymbol{x}_0^*,\dot{\boldsymbol{x}}_0^*$，该初值可以直接用于隐式微分方程求解函数 `ode15i()`，该函数的返回变量 `res.x` 和 `res.y` 将分别为 $\boldsymbol{t}$ 和 $\boldsymbol{x}$。下面将通过具体例子演示隐式微分方程的求解方法。

【例 7-19】试用隐式微分方程求解的方法解出例 7-18 中给出的隐式微分方程。

【求解】选择状态变量 $x_1 = x, x_2 = \dot{x}, x_3 = y, x_4 = \dot{y}$，则原方程可以变换成

$$\begin{cases} \dot{x}_1 - x_2 = 0 \\ \dot{x}_2 \sin x_4 + \dot{x}_4^2 + 2x_1x_3 - x_1\dot{x}_2x_4 = 0 \\ \dot{x}_3 - x_4 = 0 \\ x_1\dot{x}_2\dot{x}_4 + \cos\dot{x}_4 - 3x_3x_2 = 0 \end{cases}$$

这样，可以编写出如下所示的描述隐式微分方程的 MATLAB 函数：

```
function fun=c7mimp(t,x,xd)
fun=[xd(1)-x(2);
     xd(2)*sin(x(4))+xd(4)^2+2*x(1)*x(3)-x(1)*xd(2)*x(4);
     xd(3)-x(4);
     x(1)*xd(2)*xd(4)+cos(xd(4))-3*x(3)*x(2)];
```

其中，初值 $\boldsymbol{x}_0$ 与前面例子中一致。由于 $\dot{\boldsymbol{x}}$ 与 $\boldsymbol{x}$ 直接的关系，可以直接表示出 $\dot{\boldsymbol{x}}_0 = [0; 0; 1; 1]^{\mathrm{T}}$，调用下面的语句即可以求解隐式微分方程，且可绘制出该方程解的时间曲线，和图 7-13 中给出的曲线完全一致。

```
>> x0=[1;0;0;1]; xd0=[0;0;1;1];  % 定义 x0 和 ẋ0
   res=ode15i('c7mimp',[0,2],x0,xd0);  % 求解隐式微分方程
   plot(res.x,res.y)  % 绘制各个状态的时间响应曲线
```

7.3.3 微分代数方程的求解

在前面的介绍中，所介绍的常微分方程数值解法主要是针对能够转换成一阶常微分方程组的类型，假设其中的一些微分方程退化为代数方程，则用前面介绍的算法无法求解，必须借助微分代数方程的特殊解法。

所谓微分代数方程 (differential algebraic equation，DAE)，是指在微分方程中，某些变量间满足某些代数方程的约束，所以这样的方程不能用前面介绍的常微分方程解法直接进行求解。假设微分方程的更一般形式可以写成

$$\boldsymbol{M}(t,\boldsymbol{x})\dot{\boldsymbol{x}} = \boldsymbol{f}(t,\boldsymbol{x}) \tag{7-3-2}$$

描述 $\boldsymbol{f}(t,\boldsymbol{x})$ 的方法和普通常微分方程完全一致，而对真正的微分代数方程来说，$\boldsymbol{M}(t,\boldsymbol{x})$ 矩阵为奇异矩阵，在微分代数方程求解程序中应该由求解选项中的成员变量 `Mass` 来表示该矩阵，考虑了这些因素则可以立即求解方程的解了。

【例 7-20】考虑下面给出的微分代数方程

$$\begin{cases} \dot{x}_1 = -0.2x_1 + x_2x_3 + 0.3x_1x_2 \\ \dot{x}_2 = 2x_1x_2 - 5x_2x_3 - 2x_2^2 \\ x_1 + x_2 + x_3 - 1 = 0 \end{cases}$$

并已知初始条件为 $x_1(0) = 0.8,\ x_2(0) = x_3(0) = 0.1$，试求取该方程的数值解。

【求解】可以看出，最后的一个方程为代数方程，可以视之为 3 个状态变量间的约束关系。用矩阵的形式可以表示该微分代数方程为

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -0.2x_1 + x_2x_3 + 0.3x_1x_2 \\ 2x_1x_2 - 5x_2x_3 - 2x_2^2 \\ x_1 + x_2 + x_3 - 1 \end{bmatrix}$$

这样就可以写出相应的 MATLAB 函数如下：

```
function dx=c7eqdae(t,x)
dx=[-0.2*x(1)+x(2)*x(3)+0.3*x(1)*x(2);
     2*x(1)*x(2)-5*x(2)*x(3)-2*x(2)*x(2);
     x(1)+x(2)+x(3)-1];
```

可以将 $\boldsymbol{M}$ 矩阵输入给 MATLAB 工作空间，并在命令窗口中给出如下命令[①]：

```
>> M=[1,0,0; 0,1,0; 0,0,0]; options=odeset; options.Mass=M;
   x0=[0.8; 0.1; 0.1]; [t,x]=ode15s(@c7eqdae,[0,20],x0,options); plot(t,x)
```

由上面的语句可以得出此微分代数方程的解，如图 7-14 所示。

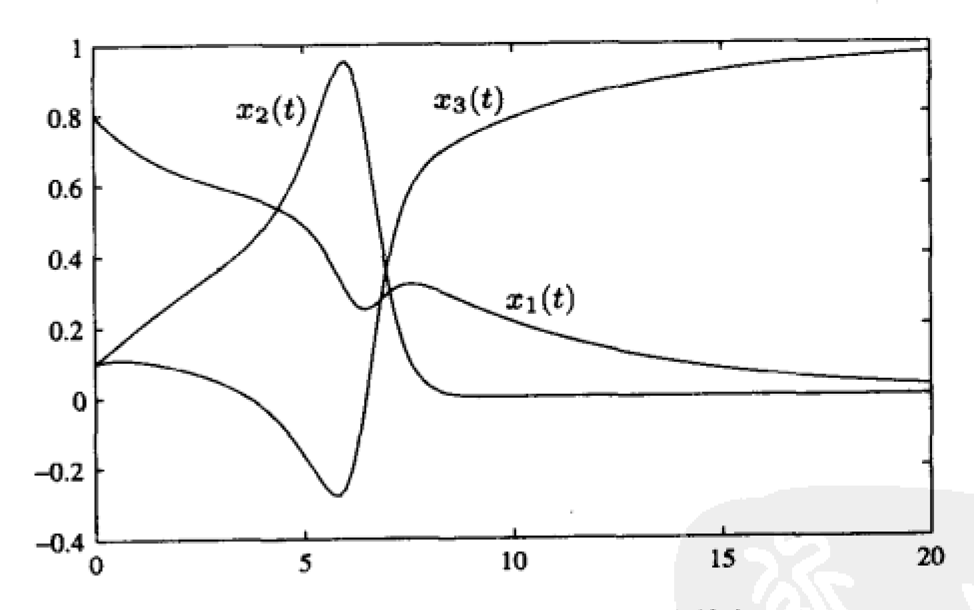


图 7-14 微分代数方程的数值解

事实上，有些微分代数方程可以转换成常微分方程求解。例如，在本例中，可以从约束式子中求出 $x_3(t) = 1 - x_1(t) - x_2(t)$，将其代入其他两个微分方程式子，则有

$$\begin{cases} \dot{x}_1 = -0.2x_1 + x_2[1 - x_1(t) - x_2(t)] + 0.3x_1x_2 \\ \dot{x}_2 = 2x_1x_2 - 5x_2[1 - x_1(t) - x_2(t)] - 2x_2^2 \end{cases}$$

根据该方程可以写出下面的 M-函数描述微分方程。

①此问题如果采用 `ode45()` 函数则将得出错误结果。

```
function dx=c7eqdae1(t,x)
dx=[-0.2*x(1)+x(2)*(1-x(1)-x(2))+0.3*x(1)*x(2);
    2*x(1)*x(2)-5*x(2)*(1-x(1)-x(2))-2*x(2)*x(2)];
```

也可以用 inline() 函数描述上述的微分方程，这样则可以用下面的命令求解变换后的微分方程组，从而最终得出原微分代数方程的解，所得出的解与前面的直接解法得出的完全一致。

```
>> x0=[0.8; 0.1];
   fDae=inline(['[-0.2*x(1)+x(2)*(1-x(1)-x(2))+0.3*x(1)*x(2);',...
        '2*x(1)*x(2)-5*x(2)*(1-x(1)-x(2))-2*x(2)*x(2)]'],'t','x');
   [t1,x1]=ode45(fDae,[0,20],x0); plot(t1,x1,t1,1-sum(x1'))
```

注意，这里如果使用 ode45() 函数也不会出现求解错误。

如果利用 MATLAB 7.0 版本通过的隐式微分方程求解函数 ode15i()，则可以用下面的语句描述隐式微分方程：

```
function fun=c7midae(t,x,xd)
fun=[xd(1)+0.2*x(1)-x(2)*x(3)-0.3*x(1)*x(2);
     xd(2)-2*x(1)*x(2)+5*x(2)*x(3)+2*x(2)^2;
     x(1)+x(2)+x(3)-1];
```

令 $x_0=[1,1,*]^{\mathrm{T}}$，其中 * 表示自由值，则可以用下面的语句解出相容的初始条件，并直接求解该微分代数方程，得出的结果将和前面完全一致，但求解更直观。

```
>> x0=[0.8;0.1;2]; x0F=[1;1;0]; xd0=[1;1;1]; xd0F=[];
   [x0,xd0]=decic(@c7midae,0,x0,x0F,xd0,xd0F); [x0,xd0]  % 相容初始条件
ans =
   0.80000000000000  -0.12600000000000
   0.10000000000000   0.09000000000000
   0.10000000000000   1.00000000000000
>> res=ode15i('c7midae',[0,20],x0,xd0); plot(res.x,res.y)
```

【例 7-21】试用微分代数方程求解的方式求解例 7-17 中定义的隐式微分方程。

【求解】在例 7-17 中采用对 $A(x)$ 矩阵直接求逆的形式原隐式方程转换成显式一阶微分方程组，这样就可以用一般微分方程组数值解法直接得出方程的解。其实，在这样的求解过程作了一个假设为 $A(x)$ 矩阵为非奇异矩阵，虽然对这个例子碰巧是正确的，但这种解法毕竟不严密，所以应该需要采用微分代数方程的方法来求解该问题。

对原方程进行分析发现，可以编写一个 M-函数来描述微分方程如下：

```
function dy=c7fimp2(t,x)
dy=[1-x(1); -x(2)];
```

注意，可能是 MATLAB 本身的漏洞，该函数不支持 inline() 形式定义的函数。在函数引用时只能采用函数句柄的形式，即 @ 后面跟函数名的形式，而不能采用引号的形式。另外，可以建立另一个 M-函数以描述质量矩阵 $A(x)$。其具体内容如下：

```
function M=c7fmass(t,x)
M=[sin(x(1)),cos(x(2)); -cos(x(2)),sin(x(1))];
```

这里仍需要使用 M-文件的形式，这样就可以用下面的语句调用微分代数方程求解微分代数方程。

```
>> options=odeset; options.Mass=@c7eqdae2; options.RelTol=1e-6;
   [t,x]=ode45(@c7fimp2,[0,10],[0;0],options); plot(t,x)
```

得出的图形仍将和图 7-12 中的曲线完全一致。

7.3.4 延迟微分方程求解

延迟微分方程组的一般形式为

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t-\tau_1), \boldsymbol{x}(t-\tau_2), \cdots, \boldsymbol{x}(t-\tau_n)) \tag{7-3-3}$$

其中，$\tau_i \geqslant 0$ 为状态变量 $\boldsymbol{x}(t)$ 的延迟常数。

MATLAB 提供了求解这类方程的隐式 Runge-Kutta 算法 `dde23()`，可以直接求解延迟微分方程。该函数的调用格式为

sol=dde23(f_1, $\boldsymbol{\tau}$, f_2, $[t_0, t_f]$)

其中，$\boldsymbol{\tau} = [\tau_1, \tau_2, \cdots, \tau_n]$，$f_1$ 为描述延迟微分方程的 MATLAB 语言函数，f_2 为描述 $t \leqslant t_0$ 时的状态变量值的函数。如果是函数则可以为 MATLAB 语言函数，如果为常量则可以由向量直接给出。该函数返回的变量 `sol` 为结构体数据，其 `sol.x` 成员变量为时间向量 $\boldsymbol{t}$，成员变量 `sol.y` 为各个时刻的状态向量构成的矩阵 $\boldsymbol{x}$，和 `ode45()` 等返回的 $\boldsymbol{x}$ 矩阵是不一样的，它是按照行排列的，正好是该函数结果的转置矩阵。可见，该函数调用格式很不规范，期望能在后面的版本中有所改变。

【例 7-22】 假设已知延迟微分方程组为

$$\begin{cases} \dot{x}(t) = 1 - 3x(t) - y(t-1) - 0.2x^3(t-0.5) - x(t-0.5) \\ \ddot{y}(t) + 3\dot{y}(t) + 2y(t) = 4x(t) \end{cases}$$

其中，在 $t \leqslant 0$ 时，$x(t) = y(t) = \dot{y}(t) = 0$，试求出该方程的数值解。

【求解】 可见，该方程中含有 $x(t), y(t)$ 信号在 $t, t-1, t-0.5$ 时刻的值，所以需要专门的延迟微分方程求解算法和程序来求解。若想得出该方程的数值解，需要将其变换成一阶显式微分方程组。实现转换的最直观方法是引入一组状态变量 $x_1(t) = x(t), x_2(t) = y(t), x_3(t) = \dot{y}(t)$，这样可以得出下面给出的一阶微分方程组

$$\begin{cases} \dot{x}_1(t) = 1 - 3x_1(t) - x_2(t-1) - 0.2x_1^3(t-0.5) - x_1(t-0.5) \\ \dot{x}_2(t) = x_3(t) \\ \dot{x}_3(t) = 4x_1(t) - 2x_2(t) - 3x_3(t) \end{cases}$$

本方程可以定义两个时间常数 $\tau_1 = 1, \tau_2 = 0.5$。这样，由第一个方程可见，在其中需要 τ_1 延迟时间常数的是状态变量 x_2，而需要 τ_2 的状态变量是 x_1，所以应编写如下的 MATLAB 函数：

```
function dx=c7exdde(t,x,z)
xlag1=z(:,1);     % 第 1 列表示提取 x(τ1)
xlag2=z(:,2);
```

```
dx=[1-3*x(1)-xlag1(2)-0.2*xlag2(1)^3-xlag2(1);
    x(3);
    4*x(1)-2*x(2)-3*x(3)];
```

历史数据函数可以如下写出：

```
function S=c7exhist(t)
S=zeros(3,1);
```

当然，由于该方程为常数初始值，所以可以直接在调用语句中使用零矩阵。这样，用下面的 MATLAB 语句可以立即得出该延迟微分方程的数值解为

```
>> lags=[1 0.5]; tx=dde23('c7exdde',lags,zeros(3,1),[0,10]);
   plot(tx.x,tx.y(2,:))
```

用下面的语句可以绘制出 $y(t)$ 信号，如图 7-15 所示。

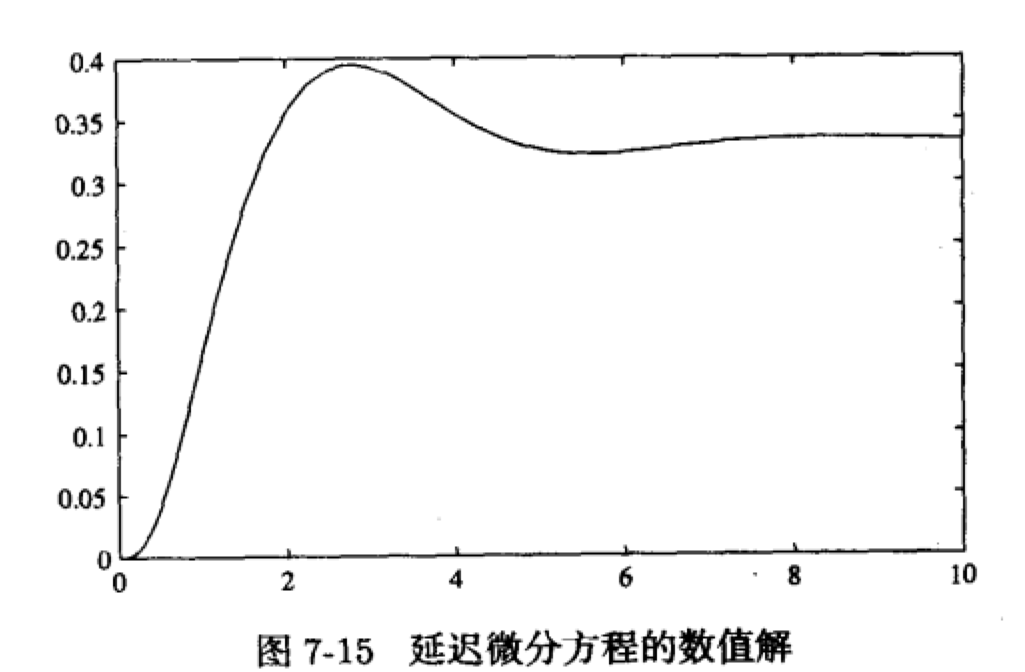


图 7-15 延迟微分方程的数值解

【例 7-23】 前面介绍了 dde23() 函数在求解延迟微分方程中的应用。然而对于如下问题

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_1\boldsymbol{x}(t-\tau_1) + \boldsymbol{A}_2\dot{\boldsymbol{x}}(t-\tau_2) + \boldsymbol{B}u(t)$$

其中，$\tau_1 = 0.15, \tau_2 = 0.5$，因为方程中同时包含 $\dot{\boldsymbol{x}}(t)$ 和 $\dot{\boldsymbol{x}}(t-\tau_2)$ 项，这类微分方程又称为中性 (neutral-type) 延迟微分方程，所以单纯采用 dde23() 是无能为力的，而需要引入更好的工具，如系统仿真的框图化求解环境 Simulink 来实现。详细的求解方法将在第 7.6 节中介绍。

7.4 边值问题的计算机求解

前面的微分方程数值解中侧重研究初值问题，即已知 $\boldsymbol{x}_0$ 对其他时刻状态变量值进行求解的方法。在实际应用中，经常会遇到这样的问题，已知部分状态在 $t=0$ 时刻的值，还知道部分状态在 $t=t_f$ 时刻的值，这类问题即所谓的边值问题。边值问题也是 ode45() 类函数无法直接求解的一类问题。本节将讨论边值问题的计算机求解方法。

二阶微分方程的边值问题的数学描述为

$$\ddot{y}(x) = F(x, y, \dot{y}) \tag{7-4-1}$$

假设想在区间 $[a,b]$ 上研究该方程的解，且已知在这两个边界点上满足

$$\alpha_a y(a)+\beta_a \dot{y}(a)=\gamma_a,\quad \alpha_b y(b)+\beta_b \dot{y}(b)=\gamma_b \tag{7-4-2}$$

上面的方程称为边界条件。

显然，利用前面的初值问题算法是不能直接使用的，因为并不能直接获得在初始时刻的各个变量的值。下面将讨论各种边值问题的数值解法。

7.4.1 线性方程边值问题的打靶算法

首先考虑一种简单的情况，即线性微分方程的边值问题求解的数值算法。考虑下面给出的微分方程

$$\ddot{y}(x)+p(x)\dot{y}(x)+q(x)y(x)=f(x) \tag{7-4-3}$$

其中，$p(x)$, $q(x)$ 和 $f(x)$ 均为给定函数，则式 (7-4-2) 可以进一步简化成

$$y(a)=\gamma_a,\ \ y(b)=\gamma_b \tag{7-4-4}$$

下面介绍的算法又称为打靶算法 (shooting method)，其基本想法是找出能够满足式 (7-4-2) 边值的相应初值 $y(0)$ 和 $\dot{y}(0)$，然后再利用前面介绍的初值算法去求解这一初值问题。打靶算法可以由下面步骤完成：

① 求出下面方程初值问题的数值解 $y_1(b)$

$$\ddot{y}_1(x)+p(x)\dot{y}_1(x)+q(x)y_1(x)=0,\ \ y_1(a)=1,\ \dot{y}_1(a)=0 \tag{7-4-5}$$

② 求出下面方程初值问题的数值解 $y_2(b)$

$$\ddot{y}_2(x)+p(x)\dot{y}_2(x)+q(x)y_2(x)=0,\ \ y_2(a)=0,\ \dot{y}_2(a)=1 \tag{7-4-6}$$

③ 求出下面方程初值问题的数值解 $y_p(b)$

$$\ddot{y}_p(x)+p(x)\dot{y}_p(x)+q(x)y_p(x)=f(x),\ \ y_p(a)=0,\ \dot{y}_p(a)=1 \tag{7-4-7}$$

④ 若 $y_2(b)\neq 0$，则计算

$$m=\frac{\gamma_b-\gamma_a y_1(b)-y_p(b)}{y_2(b)} \tag{7-4-8}$$

⑤ 计算下面初值问题的数值解，则 $y(x)$ 即为原边值问题的数值解

$$\ddot{y}(x)+p(x)\dot{y}(x)+q(x)y(x)=f(x),\ \ y(a)=\gamma_a,\ \dot{y}(a)=m \tag{7-4-9}$$

求解前面问题的数值解应该首先得出一阶微分方程组模型，即取变量 $x_1=y$, $x_2=\dot{y}$，则可以得出

$$\frac{\mathrm{d}x_1}{\mathrm{d}x}=x_2,\quad \frac{\mathrm{d}x_2}{\mathrm{d}x}=-q(x)x_1-p(x)x_2+f(x) \tag{7-4-10}$$

上面算法的 MATLAB 实现为

```
function [t,y]=shooting(f1,f2,tspan,x0f,varargin)
t0=tspan(1); tfinal=tspan(2); ga=x0f(1); gb=x0f(2);
[t,y1]=ode45(f1,tspan,[1;0],varargin);
[t,y2]=ode45(f1,tspan,[0;1],varargin);
[t,yp]=ode45(f2,tspan,[0;0],varargin);
m=(gb-ga*y1(end,1)-yp(end,1))/y2(end,1);
[t,y]=ode45(f2,tspan,[ga;m],varargin);
```

在该函数中，tspan 为初始和终止仿真时间构成的向量，x_{0f}=[γ_a;γ_b] 为边界值。除此之外，用户需要定义两个辅助函数 f1(), f2() 来描述原模型。其中，f1() 描述

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -q(x_1)x_1 - p(x_1)x_2 + f(x_1) \end{cases} \tag{7-4-11}$$

而 f2() 描述原来方程的齐次部分

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -q(x_1)x_1 - p(x_1)x_2 \end{cases} \tag{7-4-12}$$

可以看出，在求解其中的初值问题时直接采用了 ode45() 函数，并允许使用该函数的任何附加选项，这些选项可以由 varargin 函数传给 ode45() 函数。如果您觉得 ode45() 函数不适合某些特定的问题，则可以用其他函数名取代之。下面将通过简单例子介绍该函数的使用。

【例 7-24】 试求线性微分方程 $\ddot{y} - 3\dot{y} + 2y = x$, $y(0) = 1, y(1) = 2$ 在 $[0, 1]$ 段方程的数值解。

【求解】 由给出的方程可以立即写出下面的两个 M 函数，其中 c7fun1.m 为

```
function xdot=c7fun1(t,x)
xdot=[x(2); -2*x(1)+3*x(2)];
```

而 c7fun2.m 为

```
function xdot=c7fun2(t,x)
xdot=[x(2); t-2*x(1)+3*x(2)];
```

采用下面的语句就可以求出原方程的解，并将其绘制出来，如图 7-16 所示。可见，得出的 $x_1(t)$ 函数满足给定的边界条件。

```
>> [t,y]=shooting('c7fun1','c7fun2',[0,1],[1;2]); plot(t,y)
```

其实，对这样简单的问题还可以由微分方程理论得出原方程的解析解为

$$y(x) = \frac{(e^2-3)e^x + (3-e)e^{2x}}{4e(e-1)} + \frac{3}{4} + \frac{1}{2}x$$

可以由下面的语句求出上面得出结果的精度。

```
>> y0=((exp(2)-3)*exp(t)+(3-exp(1))*exp(2*t))/(4*exp(1)*(exp(1)-1))+3/4+t/2;
   norm(y(:,1)-y0)  % 整个解函数检验
```

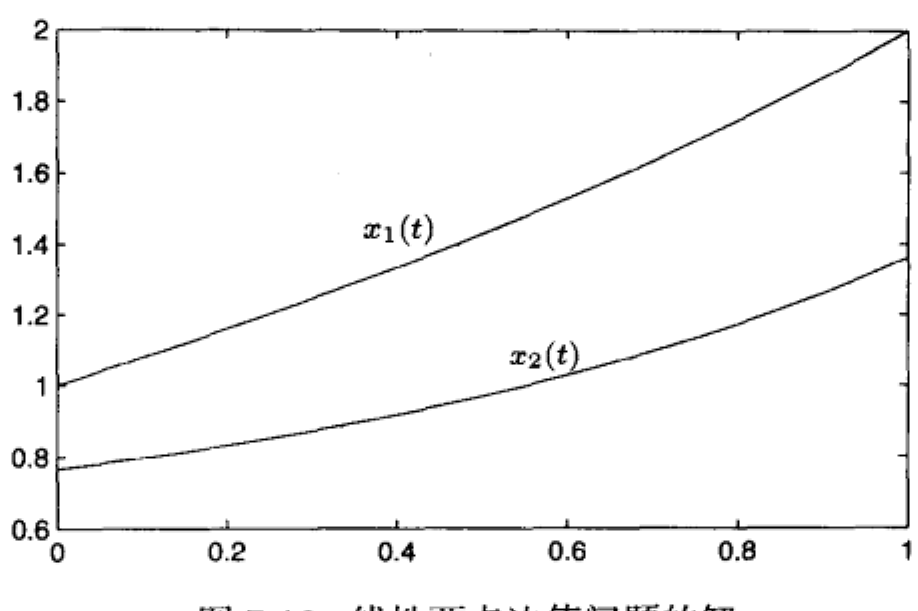


图 7-16 线性两点边值问题的解

```
ans =
     4.479031687538188e-008
>> norm(y(end,1)-2) % 终点条件检验
ans =
     2.262022391619212e-008
```

由检验结果可见，这样求出的解精度还是较高的。

7.4.2 非线性方程边值问题的打靶算法

前面对一类特殊的微分方程 (即线性微分方程) 边值问题进行了探讨，但这种方法并不能求解式 (7-4-1) 和 (7-4-2) 所描述的一般问题。这里给出对此一般问题的数值解法。

假定原始问题可以转换成下面的初值问题

$$\ddot{y} = F(x, y, \dot{y}),\ \ y(a) = \gamma_a,\ \dot{y}(a) = m \tag{7-4-13}$$

则问题转换成求解 $y(m; b) = \gamma_b$，可以采用下面的 Newton 迭代法来求取 m。

$$m_{i+1} = m_i - \frac{y(m_i; b) - \gamma_b}{(\partial y/\partial m)(m_i; b)} = m_i - \frac{v_1(b) - \gamma_b}{v_3(b)} \tag{7-4-14}$$

式中，$v_1 = y(m_i; x), v_2 = \dot{y}(m_i; x), v_3 = (\partial y/\partial m)(m_i; x), v_4 = (\partial \dot{y}/\partial m)(m_i; x)$，且可以由下面的微分方程初值问题来求解。

$$\left\{\begin{array}{l} \dot{v}_1 = v_2,\ \ v_1(a) = \gamma_a \\ \dot{v}_2 = F(x, v_1, v_2),\ \ v_2(a) = m \\ \dot{v}_3 = v_4,\ \ v_3(a) = 0 \\ \dot{v}_4 = \dfrac{\partial F}{\partial y}(x, v_1, v_2)v_3 + \dfrac{\partial F}{\partial \dot{y}}(x, v_1, v_2)v_4,\ \ v_4(a) = 1 \end{array}\right. \tag{7-4-15}$$

其中，要求能显式地求出 $\partial F/\partial y$, $\partial F/\partial \dot{y}$。在具体计算中可以指定一个 m 值，然后求解式 (7-4-15) 中的初值问题，将结果代入式 (7-4-14) 迭代一步，并将结果代入式 (7-4-15) 重

新计算，直至两次计算出来的 m 值的误差在允许的范围内，则可以采用此 m 值，最后代入式 (7-4-13) 来求解原始问题。上面算法的 MATLAB 实现为

```
function [t,y]=nlbound(funcn,funcv,tspan,x0f,tol,varargin)
t0=tspan(1);tfinal=tspan(2);  ga=x0f(1); gb=x0f(2); m=1; m0=0;
while (norm(m-m0)>tol), m0=m;
   [t,v]=ode45(funcv,tspan,[ga;m;0;1],varargin);
   m=m0-(v(end,1)-gb)/(v(end,3));
end
[t,y]=ode45(funcn,tspan,[ga;m],varargin);
```

其中，用户必须自己编写一个 funcv() 函数来描述式 (7-4-15) 这的初值问题。下面将通过例子来演示此算法。

【例 7-25】 试求解下面的非线性微分方程边值问题。

$$\ddot{y} = F(x, y, \dot{y}) = 2y\dot{y},\ y(0) = -1,\ y(\pi/2) = 1$$

【求解】 可以容易地求出偏导数 $\partial F/\partial y = 2\dot{y}$, $\partial F/\partial \dot{y} = 2y$，代入式 (7-4-15) 中的第 4 个式子可以立即得出 $\dot{v}_4 = 2x_2x_3 + 2x_1x_4$，故可以写出下面的 MATLAB 函数为

```
function xdot=c7fun3(t,x)
xdot=[x(2); 2*x(1)*x(2); x(4); 2*x(2)*x(3)+2*x(1)*x(4)];
```

而原微分方程模型的 MATLAB 表示可以写成

```
function xdot=c7fun4(t,x)
xdot=[x(2); 2*x(1)*x(2)];
```

这时，可以通过下面的 MATLAB 语句来求解原始问题，得出的结果如图 7-17 所示。可见，解出的 $x_1(t)$ 函数满足给定的边界要求。

```
>> [t,y]=nlbound('c7fun4','c7fun3',[0,pi/2],[-1,1],1e-8);
   plot(t,y); set(gca,'xlim',[0,pi/2]);
```

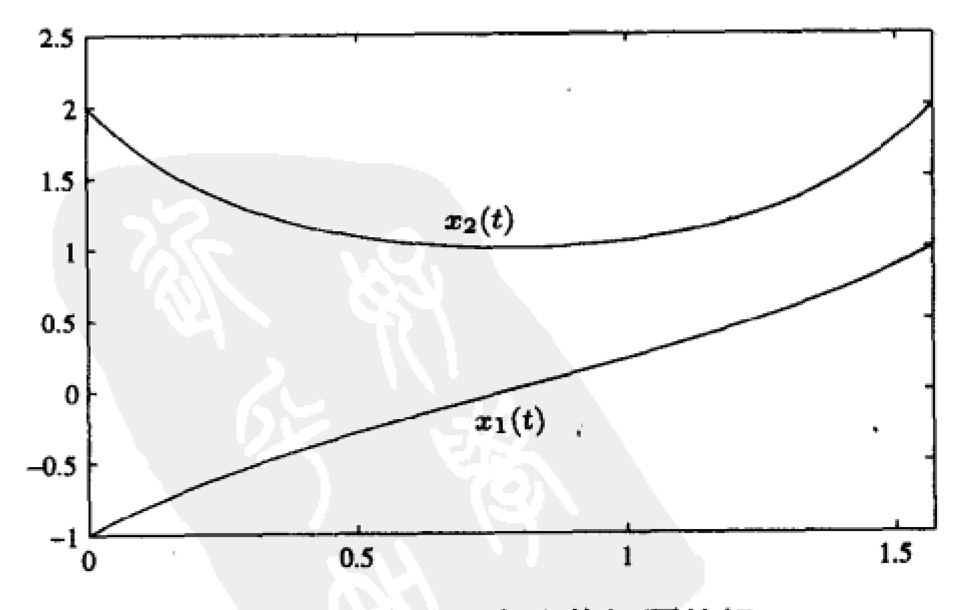


图 7-17 线性两点边值问题的解

该问题的解析解为 $y(x) = \tan(x - \pi/4)$。这样，就可以由下面的语句比较得出的解是否精确

了。

```
>> y0=tan(t-pi/4); norm(y(:,1)-y0)
ans =
    1.662871328177840e-005
>> norm(y(end,1)-1)
ans =
    5.281529774414651e-006
```

可见，这样得出的数值解和解析解极其接近，且满足原始边值条件。

7.4.3 线性微分方程的有限差分算法

考虑式 (7-4-3) 中给出的线性微分方程，如果用前面介绍的中心差分数值微分公式取代其中的微分项，则可以推导出下面的线性代数方程组

$$\begin{bmatrix} t_1 & v_1 & & & \\ w_2 & t_2 & v_2 & & \\ & \ddots & \ddots & \ddots & \\ & & w_{n-2} & t_{n-2} & v_{n-2} \\ & & & w_{n-1} & t_{n-1} \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_{n-2} \\ \eta_{n-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix} \tag{7-4-16}$$

其中，n 为中间计算点的个数，$\eta_i, i=1,2,\cdots,n-1$ 为微分方程的数值解，即 $y_{i+1}=\eta_i$，而 $y_1=\gamma_a$，$y_n=\gamma_b$，其他参数可以由下式计算出来，为

$$\begin{aligned} & t_i=-2+h^2 q(x_i),\ v_i=1+\frac{h}{2}p(x_i) \\ & w_i=1-\frac{h}{2}p(x_i),\ b_i=h^2 f(x_i) \end{aligned} \quad i=1,\cdots,n-1 \tag{7-4-17}$$

且

$$b_1=b_1-w_1\gamma_a,\quad b_{n-1}=b_{n-1}-v_{n-1}\gamma_b \tag{7-4-18}$$

式中，h 为计算步长，且 $h=(b-a)/(n-1)$, $x_i=a+(i-1)h$。上面算法的 MATLAB 实现为

```
function [x,y]=fdiff(funcs,tspan,x0f,n)
t0=tspan(1);tfinal=tspan(2); ga=x0f(1); gb=x0f(2);
h=(tfinal-t0)/n;
for i=1:n, x(i)=t0+h*(i-1); end,
x0=x(1:n-1); t=-2+h^2*feval(funcs,x0,2); tmp=feval(funcs,x0,1);
v=1+h*tmp/2; w=1-h*tmp/2; b=h^2*feval(funcs,x0,3);
b(1)=b(1)-w(1)*ga; b(n-1)=b(n-1)-v(n-1)*gb; b=b'; A=diag(t);
for i=1:n-2, A(i,i+1)=v(i); A(i+1,i)=w(i+1); end
y=inv(A)*b;  x=[x tfinal]; y=[ga; y; gb]';
```

其中，要求给定的函数为 funcs(x,key)，当 key=1,2,3 时分别表示 $p(x),q(x),f(x)$ 函数。另外，n 为要计算的点数。从算法来看，要求等间距的计算点分布，所以在解决一些复杂问题时，该算法不一定实用。这里将通过例子来演示该函数的使用。

【例 7-26】试求解线性微分方程 $\ddot{y}+(1+x)\dot{y}+(1-x)y=1+x^2, y(0)=1, y(1)=4$ 的边值问题。

【求解】对该微分方程可以编写下面的 MATLAB 函数:

```
function y=c7fun5(x,key)
switch key
case 1, y=1+x;
case 2, y=1-x;
otherwise, y=1+x.^2;
end
```

这时要计算出各个计算点处的 y 值，则可以由下面的方式调用 fdiff() 函数，绘制出 $y(t)$ 的曲线，如图 7-18 所示。

```
>> [t,y]=fdiff('c7fun5',[0,1],[1,4],50); plot(t,y)
```

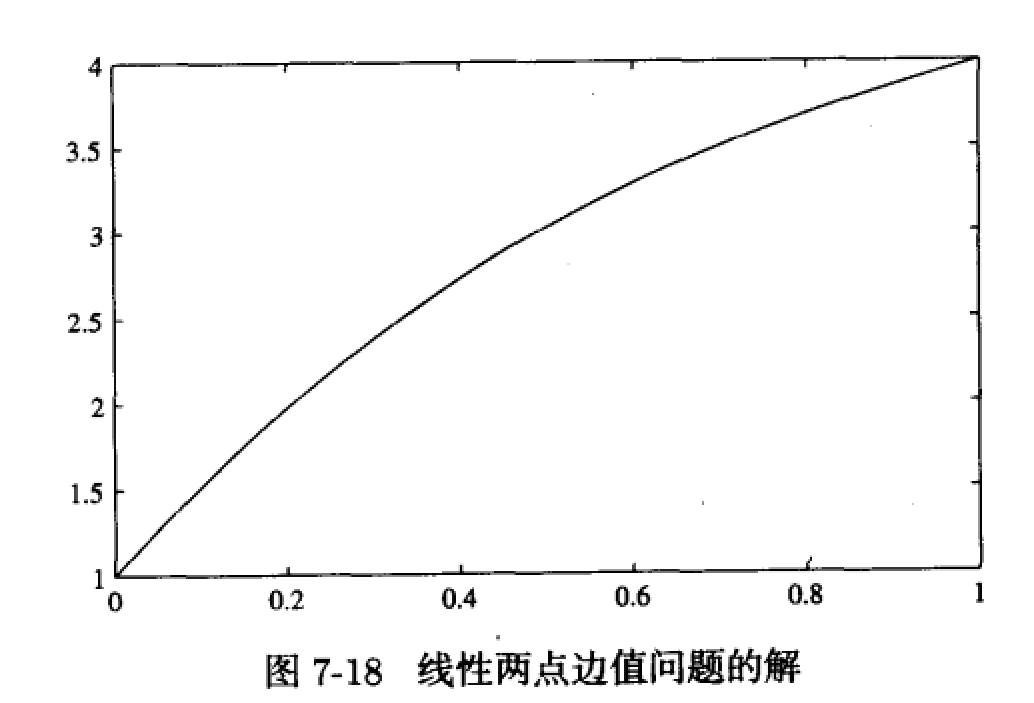


图 7-18 线性两点边值问题的解

当然，利用式 (7-4-16) 方程的三对角特性，可以采用 LU 分解来容易地求解，但这样会使得 MATLAB 编程变得更烦琐。从前面给出的算法看，只有第 2 种算法不要求原始函数为线性的，但该算法需要求出 $F(\cdot)$ 函数的偏导数，所以一般在线性方程求解时采用第 1 种方法，其他情况下可以采用第 2 种方法。

7.5 偏微分方程求解入门

MATLAB 可以求解一般的偏微分方程，也可以用偏微分方程工具箱中给出的相应函数求解一些偏微分方程。本节将首先介绍一般偏微分方程的数值解法，然后介绍利用偏微分方程工具箱求解几类典型偏微分方程的方法。

7.5.1 偏微分方程组求解

MATLAB 语言提供了 pdepe() 函数，可以直接求解偏微分方程

$$\boldsymbol{c}\left(x,t,\boldsymbol{u},\frac{\partial \boldsymbol{u}}{\partial x}\right)\frac{\partial \boldsymbol{u}}{\partial t}=x^{-m}\frac{\partial}{\partial x}\left[x^m\boldsymbol{f}\left(x,t,\boldsymbol{u},\frac{\partial \boldsymbol{u}}{\partial x}\right)\right]+\boldsymbol{s}\left(x,t,\boldsymbol{u},\frac{\partial \boldsymbol{u}}{\partial x}\right) \tag{7-5-1}$$

这样，偏微分方程可以编写下面的函数描述，其入口为

$[\boldsymbol{c},\boldsymbol{f},\boldsymbol{s}]$=pdefun($x,t,\boldsymbol{u},\boldsymbol{u}_x$)

其中，pdefun 为函数名。这样，由给定的输入变量即可计算出 $\boldsymbol{c},\boldsymbol{f},\boldsymbol{s}$ 这 3 个函数。

边界条件可以用下面的函数描述为

$$\boldsymbol{p}(x,t,\boldsymbol{u})+\boldsymbol{q}(x,t,\boldsymbol{u}).*\boldsymbol{f}\left(x,t,\boldsymbol{u},\frac{\partial \boldsymbol{u}}{\partial x}\right)=\boldsymbol{0} \tag{7-5-2}$$

这样的边值函数可以编写一个如下的 MATLAB 函数描述为

$[\boldsymbol{p}_a,\boldsymbol{q}_a,\boldsymbol{p}_b,\boldsymbol{q}_b]$=pdebc($x,t,\boldsymbol{u},\boldsymbol{u}_x$)

除了这两种函数外，还应该写出初始条件函数。偏微分方程初始条件的数学描述为 $\boldsymbol{u}(x,t_0)=\boldsymbol{u}_0$。这样，需要一个简单的函数来描述，编写简单函数 u_0=pdeic(x) 即可。

还可以选择 $\boldsymbol{x}$ 和 $\boldsymbol{t}$ 的向量，再加上描述的这些函数，就可以用 pdepe() 函数求解次偏微分方程，这需要用下面的格式求解该偏微分方程。

sol=pdepe(m,@pdefun,@pdeic,@pdebc,$\boldsymbol{x}$,$\boldsymbol{t}$)

【例 7-27】 试求解下面的偏微分方程[27]

$$\begin{cases}\dfrac{\partial u_1}{\partial t}=0.024\dfrac{\partial^2 u_1}{\partial x^2}-F(u_1-u_2)\\[2mm]\dfrac{\partial u_2}{\partial t}=0.17\dfrac{\partial^2 u_2}{\partial x^2}+F(u_1-u_2)\end{cases}$$

其中，$F(x)=\mathrm{e}^{5.73x}-\mathrm{e}^{-11.46x}$，且满足初始条件 $u_1(x,0)=1,\ u_2(x,1)=0$ 及边界条件

$$\frac{\partial u_1}{\partial x}(0,t)=0,\ u_2(0,t)=0,\ u_1(1,t)=1,\ \frac{\partial u_2}{\partial x}(1,t)=0$$

【求解】 对照给出的偏微分方程和式 (7-5-1)，则可以将原方程改写为

$$\begin{bmatrix}1\\1\end{bmatrix}.*\frac{\partial}{\partial t}\begin{bmatrix}u_1\\u_2\end{bmatrix}=\frac{\partial}{\partial x}\begin{bmatrix}0.024\partial u_1/\partial x\\0.17\partial u_2/\partial x\end{bmatrix}+\begin{bmatrix}-F(u_1-u_2)\\F(u_1-u_2)\end{bmatrix}$$

可见，$m=0$，且

$$\boldsymbol{c}=\begin{bmatrix}1\\1\end{bmatrix},\ \boldsymbol{f}=\begin{bmatrix}0.024\partial u_1/\partial x\\0.17\partial u_2/\partial x\end{bmatrix},\ \boldsymbol{s}=\begin{bmatrix}-F(u_1-u_2)\\F(u_1-u_2)\end{bmatrix}$$

这样，可以编写出下面的描述偏微分方程的 MATLAB 函数为

```
function [c,f,s]=c7mpde(x,t,u,du)
c=[1;1]; y=u(1)-u(2); F=exp(5.73*y)-exp(-11.46*y); s=F*[-1; 1];
f=[0.024*du(1); 0.17*du(2)];
```

套用式 (7-5-2) 中的边界条件，可以写出如下的边值方程：

$$\text{左边界}\ \begin{bmatrix}0\\u_2\end{bmatrix}+\begin{bmatrix}1\\0\end{bmatrix}.*\boldsymbol{f}=\begin{bmatrix}0\\0\end{bmatrix},\quad \text{右边界}\ \begin{bmatrix}u_1-1\\0\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}.*\boldsymbol{f}=\begin{bmatrix}0\\0\end{bmatrix}$$

从而编写出下面描述边界条件的 MATLAB 函数。

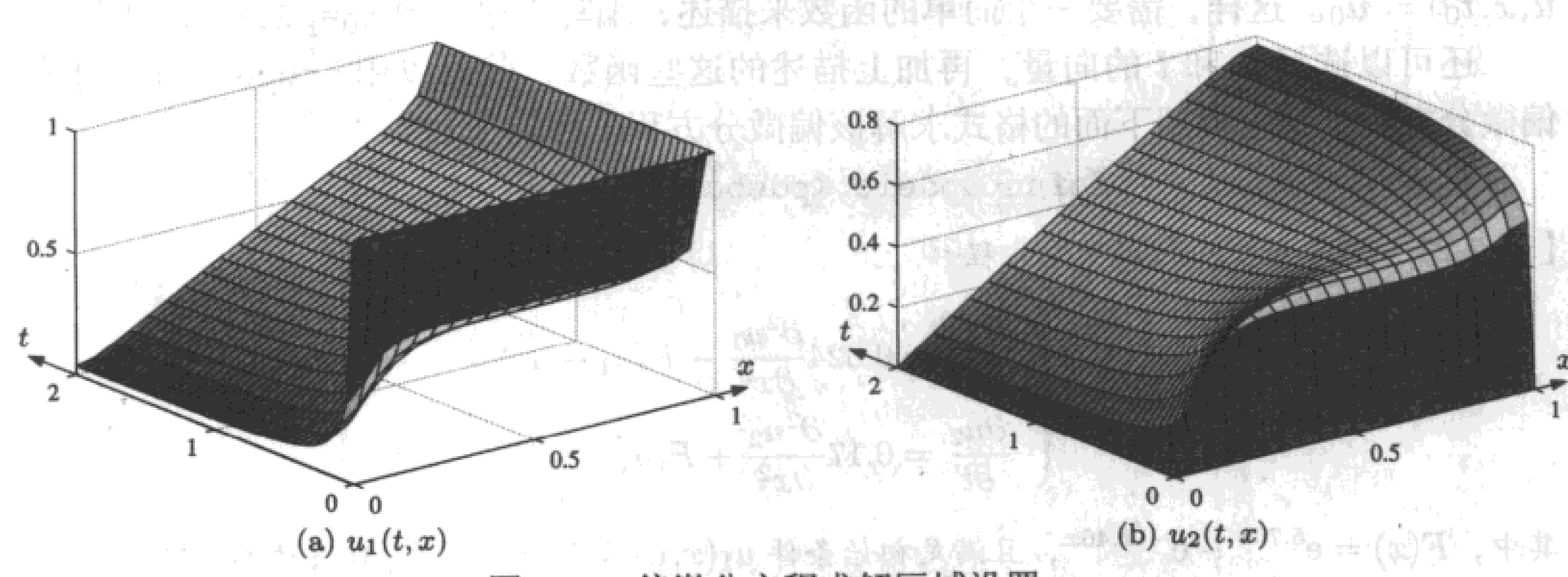
```
function [pa,qa,pb,qb]=c7mpbc(xa,ua,xb,ub,t)
pa=[0; ua(2)]; qa=[1;0]; pb=[ub(1)-1; 0]; qb=[0;1];
```

另外，还可以立即得出描述初值的 MATLAB 函数为

```
function u0=c7mpic(x)
u0=[1; 0];
```

有了这 3 个函数，选定 x 和 t 向量，则可以由下面的语句直接求解此偏微分方程，得出解 u_1 和 u_2，如图 7-19 (a)、图 7-19 (b) 所示。

```
>> x=0:.05:1; t=0:0.05:2; m=0;
   sol=pdepe(m,@c7mpde,@c7mpic,@c7mpbc,x,t); surf(x,t,sol(:,:,1))
   figure;  surf(x,t,sol(:,:,2))
```



(a) $u_1(t,x)$　　(b) $u_2(t,x)$

图 7-19 偏微分方程求解区域设置

7.5.2 二阶偏微分方程的数学描述

除了前面介绍的一类偏微分方程外，MATLAB 语言还有自己的偏微分方程工具箱，可以比较规范地求解各种常见的二阶偏微分方程。这里将 MATLAB 偏微分方程工具箱可解的二阶偏微分方程简单介绍一下，后面再介绍一个实用的偏微分方程求解界面 pdetool，利用该程序界面可以容易地求解偏微分方程。

7.5.2.1 椭圆型偏微分方程

椭圆型偏微分方程的一般表示形式为

$$-\mathrm{div}(c\nabla u)+au=f(\boldsymbol{x},t) \tag{7-5-3}$$

其中，若 $u=u(x_1,x_2,\cdots,x_n,t)=u(\boldsymbol{x},t)$，$\nabla u$ 为 u 的梯度，则其定义为

$$\nabla u=\left[\frac{\partial}{\partial x_1},\frac{\partial}{\partial x_2},\cdots,\frac{\partial}{\partial x_n}\right]u \tag{7-5-4}$$

散度 $\mathrm{div}(v)$ 的定义为

$$\mathrm{div}(v)=\left(\frac{\partial}{\partial x_1}+\frac{\partial}{\partial x_2}+\cdots+\frac{\partial}{\partial x_n}\right)v \tag{7-5-5}$$

这样，$\mathrm{div}(c\nabla u)$ 可以更明确地表示成

$$\mathrm{div}(c\nabla u)=\left[\frac{\partial}{\partial x_1}\left(c\frac{\partial u}{\partial x_1}\right)+\frac{\partial}{\partial x_2}\left(c\frac{\partial u}{\partial x_2}\right)+\cdots+\frac{\partial}{\partial x_n}\left(c\frac{\partial u}{\partial x_n}\right)\right] \tag{7-5-6}$$

若 c 为常数，则该项可以进一步化简为

$$\mathrm{div}(c\nabla u)=c\left(\frac{\partial^2}{\partial x_1^2}+\frac{\partial^2}{\partial x_2^2}+\cdots+\frac{\partial^2}{\partial x_n^2}\right)u=c\Delta u \tag{7-5-7}$$

其中，Δ 又称为 Laplace 算子。这样，椭圆型偏微分方程可以更简单地写成

$$-c\left(\frac{\partial^2}{\partial x_1^2}+\frac{\partial^2}{\partial x_2^2}+\cdots+\frac{\partial^2}{\partial x_n^2}\right)u+au=f(\boldsymbol{x},t) \tag{7-5-8}$$

7.5.2.2 抛物线型偏微分方程

抛物型偏微分方程的一般形式为

$$d\frac{\partial u}{\partial t}-\mathrm{div}(c\nabla u)+au=f(\boldsymbol{x},t) \tag{7-5-9}$$

根据上面的叙述，若 c 为常数，则该方程可以更简单地写成

$$d\frac{\partial u}{\partial t}-c\left(\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}+\cdots+\frac{\partial^2 u}{\partial x_n^2}\right)+au=f(\boldsymbol{x},t) \tag{7-5-10}$$

7.5.2.3 双曲型偏微分方程

双曲型偏微分方程的一般形式为

$$d\frac{\partial^2 u}{\partial t^2}-\mathrm{div}(c\nabla u)+au=f(\boldsymbol{x},t) \tag{7-5-11}$$

若 c 为常数，则可以将该方程简化成

$$d\frac{\partial^2 u}{\partial t^2}-c\left(\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}+\cdots+\frac{\partial^2 u}{\partial x_n^2}\right)+au=f(\boldsymbol{x},t) \tag{7-5-12}$$

从上面的 3 种类型方程可以看出，它们直接的区别在于 u 函数对 t 的导数阶次。如果对 t 没有求导，则可以理解为其值为常数，故称为椭圆型偏微分方程。如果取 u 对时间的一阶导数，则一阶导数与 u 对 $\boldsymbol{x}$ 的二阶导数直接构成了抛物线关系，故称其为抛物型偏微分方程。如果对 t 取二阶导数，则可以称之为双曲型偏微分方程。

MATLAB 的偏微分方程工具箱采用的是有限元方法求解各种偏微分方程的。椭圆型偏微分方程求解中，c,a,d,f 均可以为给定函数的形式，但其他类型偏微分方程求解时，它们必须为常数。

7.5.2.4 特征值型偏微分方程

MATLAB 可以直接求解的另一类偏微分方程为特征值型偏微分方程为

$$-\text{div}(c\nabla u)+au=\lambda du \tag{7-5-13}$$

对常数 c，该方程还可以简化成

$$-c\left(\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}+\cdots+\frac{\partial^2 u}{\partial x_n^2}\right)+au=\lambda du \tag{7-5-14}$$

对比式 (7-5-14) 与式 (7-5-3) 可以发现，将前者等号右侧的 λdu 移动到方程的左侧，就可以变换成一般的椭圆型偏微分方程，所以该方程是椭圆型偏微分方程的一个特例。

7.5.3 偏微分方程的求解界面应用举例

7.5.3.1 偏微分方程求解程序概述

MATLAB 偏微分方程工具箱提供了一个界面，可以求解二元偏微分方程 $u(x_1,x_2)$，这时求解区域可以用该界面提供的画圆、椭圆、矩形及多边形等工具任意绘制，也可以由若干个这样简单绘制的集合进行并集、交集、差集等构成所需的求解区域。完成求解区域的绘制后，还可以用该界面提供的功能将原求解区域用三角形的形式自动绘制出网格。

在 MATLAB 提示符下键入 `pdetool`，将启动偏微分方程求解界面，如图 7-20 所示。

偏微分方程求解界面分为如下几个部分：

① **菜单系统** 偏微分方程工具箱有较全面的菜单系统，其中大部分实用功能均可以由工具栏实现，工具栏不能实现的部分多为一些工具箱的设置与文件处理的功能。后面将根据实际需要介绍菜单系统的若干功能。

② **工具栏** 工具栏内各个按钮的详细内容如图 7-21 所示，工具栏能实现从求解区域设定、微分方程参数描述、求解到结果表示在内的一整套实际功能。工具栏右侧的列表框还给出了 MATLAB 能直接求解的一些常用微分方程类型。

③ **集合编辑** (Set formula) 用户可以在求解区域用不同的几何形状画出若干集合，而集合编辑区域允许用户用加减法等表示集合的并、交和差集运算，更精确地描述求解区域。

④ **求解区域** 为该程序界面下部的区域，用户可以在这个部分内绘制出问题的求解区域，微分方程的解也可以在这个区域内用二维的形式表示出来。MATLAB 还支持三维表示，但需要打开新的图形窗口。

7.5.3.2 偏微分方程求解区域绘制

本节将通过例子演示在偏微分方程求解界面下描述求解区域的方法。首先用工具栏中提供的椭圆绘制和矩形绘制功能绘制出如图 7-22 (a) 所示的一些区域，这样就可以在集合编辑栏目中将原来的内容修改为 `(R1+E1+E2)-E3`，表示从矩形 `R1`，椭圆 `E1`, `E2` 的并集中剔除掉 `E3`。单击工具栏中 $\partial\Omega$ 按钮就可以得到求解区域。选择 Boundary → Remove All

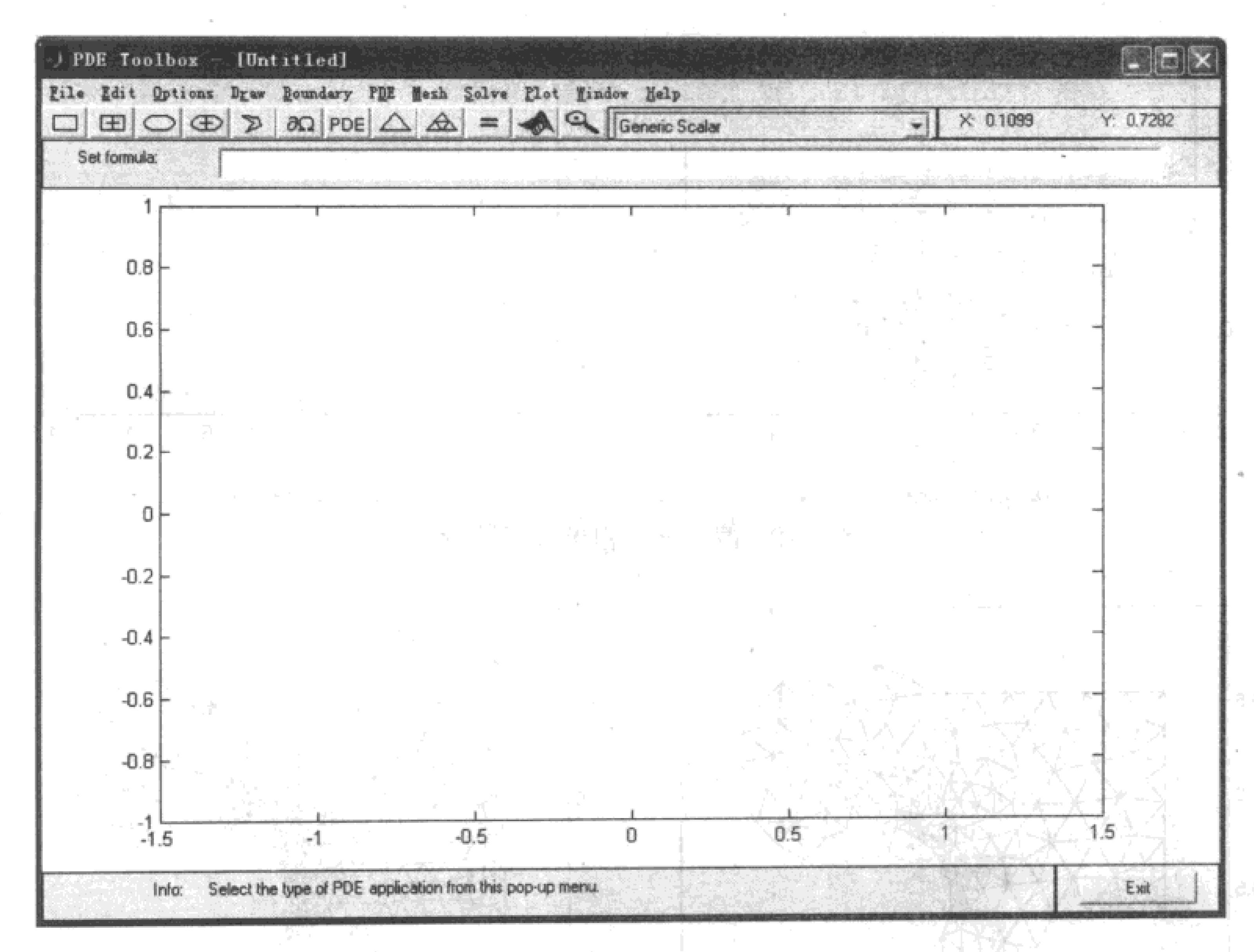


图 7-20 偏微分方程求解界面

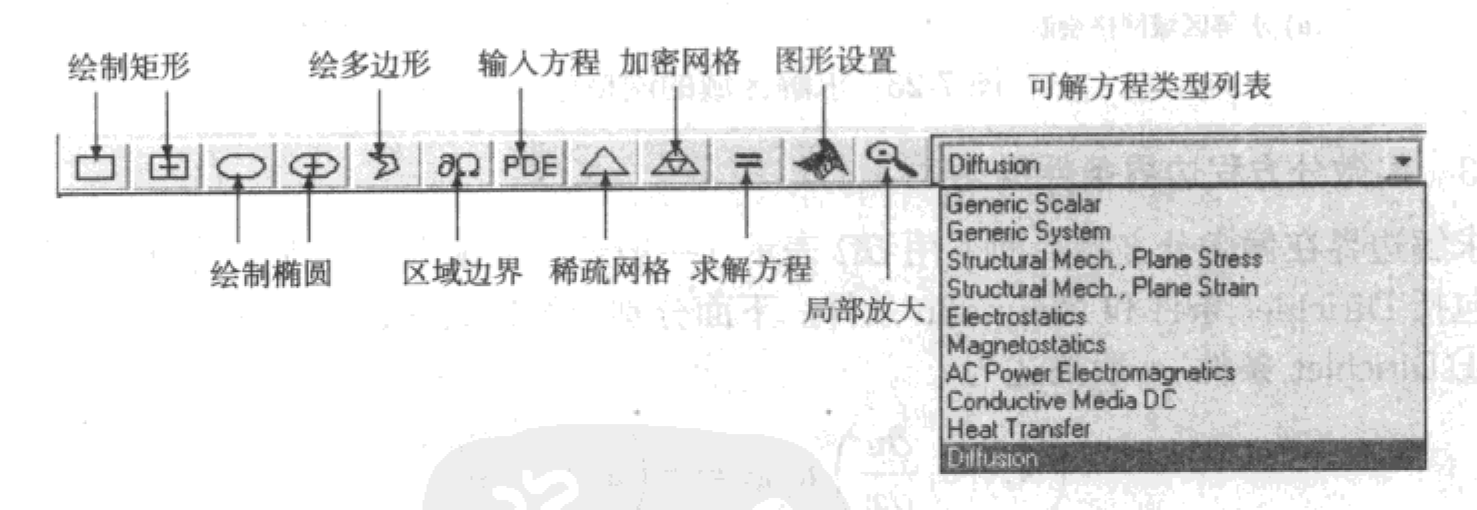
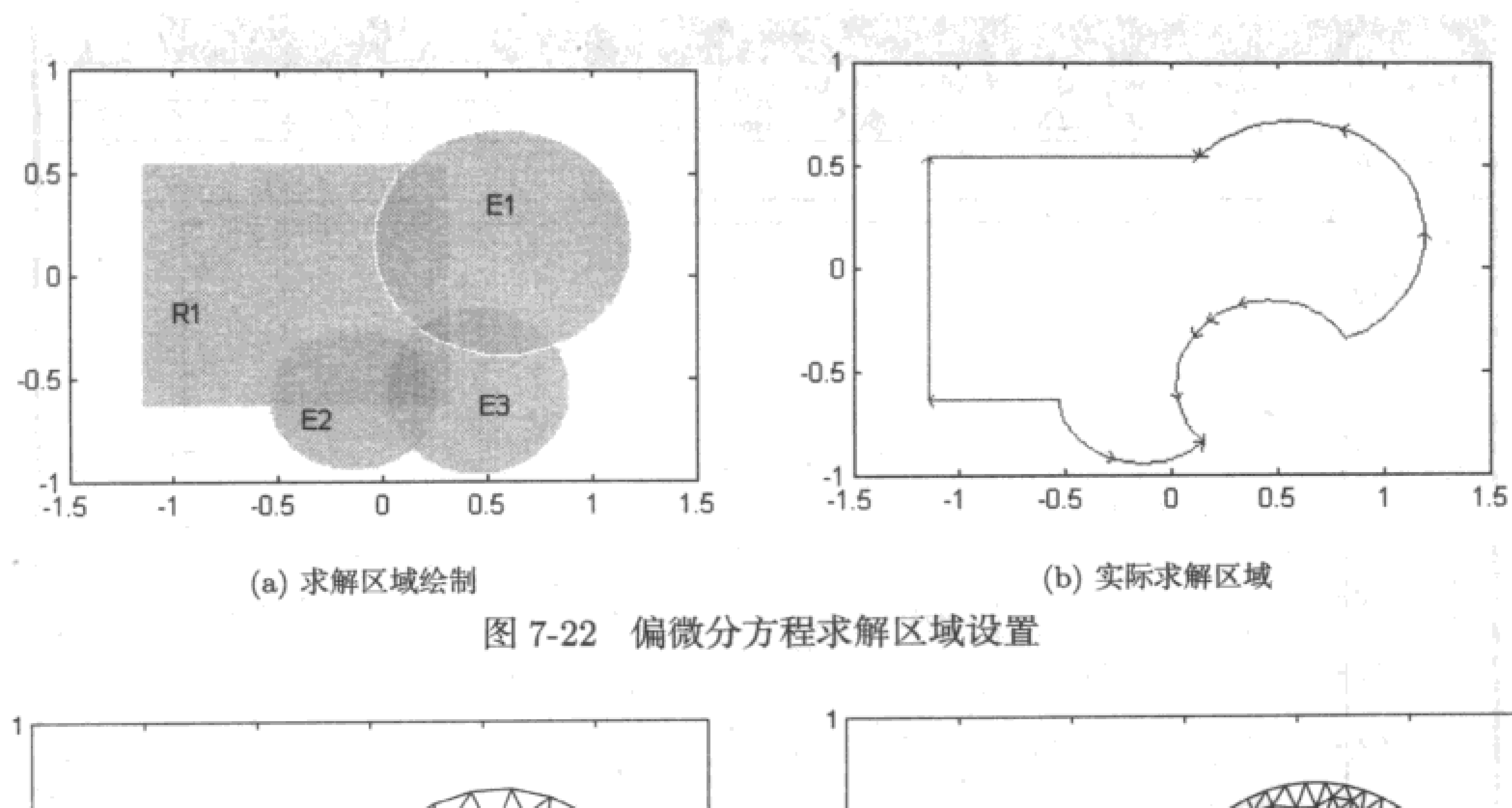


图 7-21 偏微分方程求解工具栏

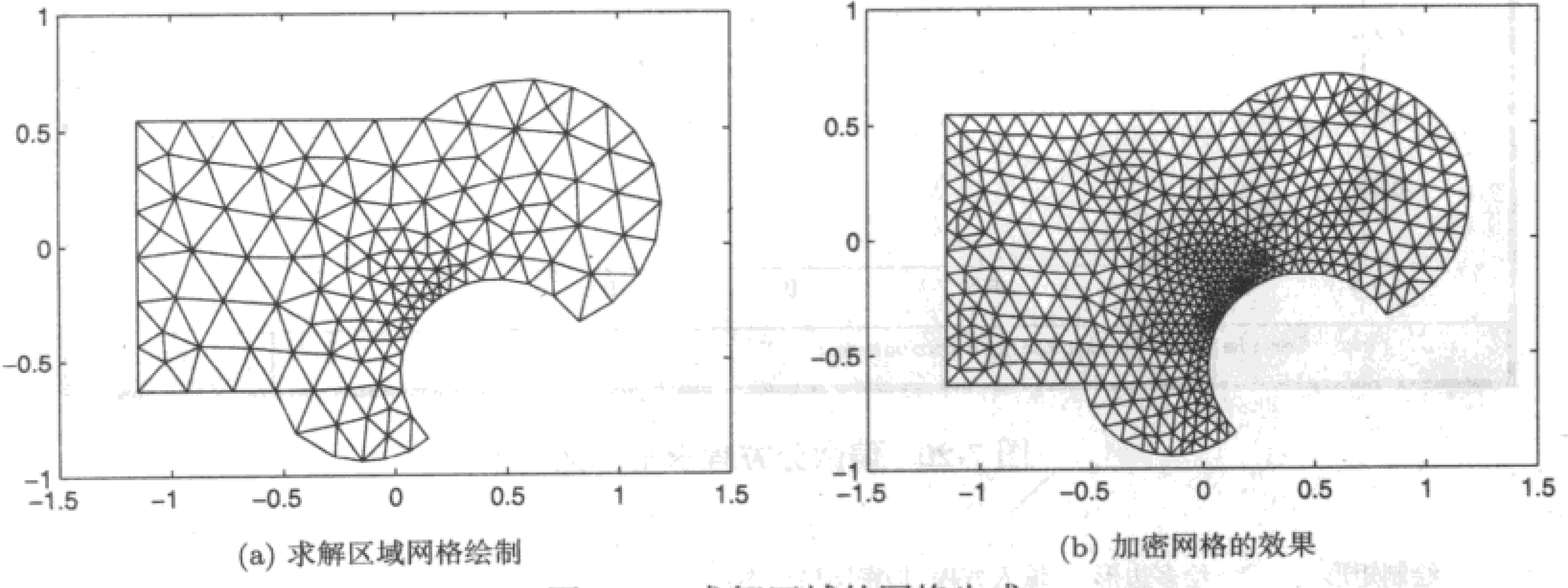
Subdomain Borders 菜单项，则将消除若干相邻区域中间的分隔线，得出如图 7-22 (b) 所示的区域图。

有了求解区域，就可以单击 Δ 按钮将求解区域用三角形划分成若干网格，如图 7-23 (a) 所示。如果感觉到网格不够密，则可以单击右侧的按钮加密网格，可以得出如图 7-23 (b) 所示的更密的网格图。值得指出的是，一般情况下，网格越密，计算的结果越精确，但代价是计算时间则越长。



(a) 求解区域绘制 (b) 实际求解区域

图 7-22 偏微分方程求解区域设置



(a) 求解区域网格绘制 (b) 加密网格的效果

图 7-23 求解区域的网格生成

7.5.3.3 偏微分方程边界条件描述

求解边界在偏微分方程界面下用 $\partial\Omega$ 表示。一般地，在偏微分方程工具箱支持的边界条件包括 Dirichlet 条件和 Neumann 条件。下面分别介绍这两种边界条件。

① Dirichlet 条件 一般描述为

$$h\left(\boldsymbol{x},t,u,\frac{\partial u}{\partial \boldsymbol{x}}\right)u|_{\partial\Omega}=r\left(\boldsymbol{x},t,u,\frac{\partial u}{\partial \boldsymbol{x}}\right) \tag{7-5-15}$$

其中，$\partial\Omega$ 表示求解区域的边界。假设在边界上满足该方程，则只需给出 r 和 h 函数即可，这两个参数可以为常数，也可以为 $\boldsymbol{x}$ 的函数，甚至可以是 $u,\partial u/\partial \boldsymbol{x}$ 的函数，为方便起见，一般可以令 $h=1$。后面将介绍 Dirichlet 边界条件的描述方式。

② Neumann 条件 其扩展形式为

$$\left[\frac{\partial}{\partial \boldsymbol{n}}(c\nabla u)+qu\right]\bigg|_{\partial\Omega}=g \tag{7-5-16}$$

其中，$\partial u/\partial \boldsymbol{n}$ 为 $\boldsymbol{x}$ 向量法向的偏导数。

选择 Boundary → Specify Boundary Conditions 菜单，将打开一个如图 7-24 所示的对话框，用户可以在这个对话框中描述边界条件。如果想使得边界上各点的函数值为 0，则可以将该对话框的 r 栏值设为 0 即可。

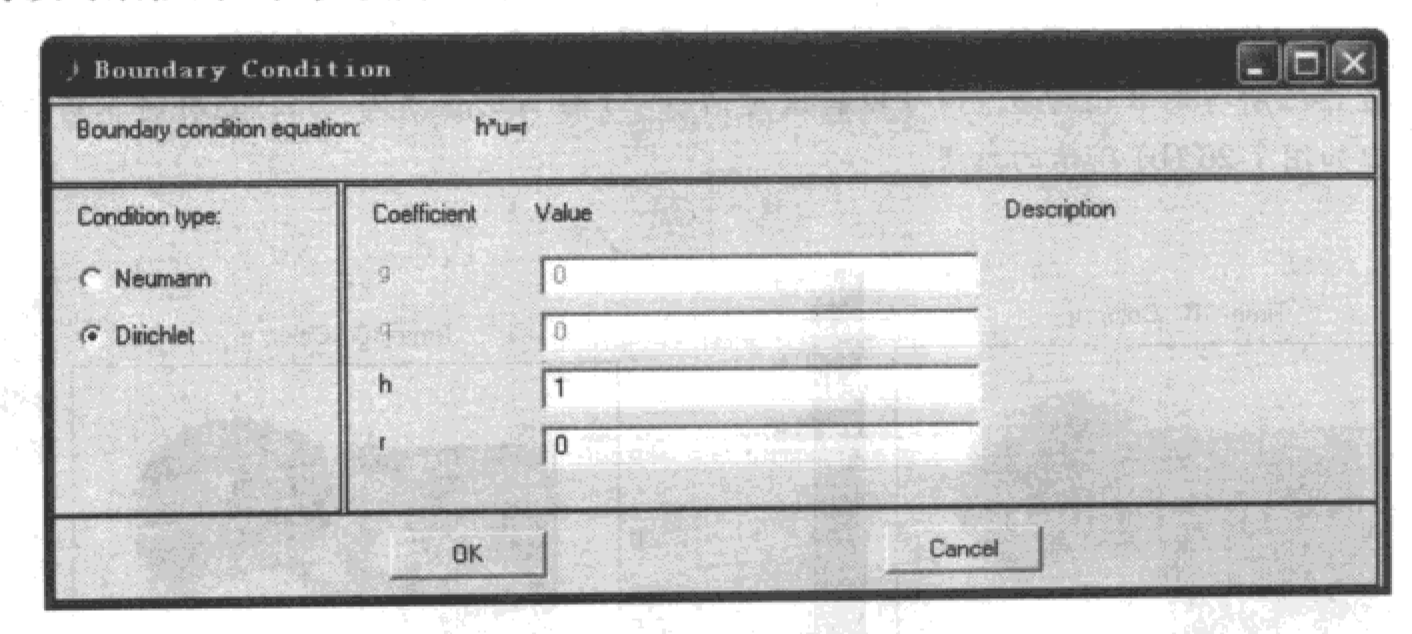


图 7-24 边界条件设置对话框

7.5.3.4 偏微分方程求解举例

用前面的方法设置了求解区域和边界条件，并选择了合适的偏微分方程后，就可以单击工具栏的等号按钮 (=) 立即得出微分方程的解。下面将通过例子演示实际偏微分方程的求解全过程。

【例 7-28】 试求解双曲型偏微分方程

$$\frac{\mathrm{d}^2u}{\mathrm{d}t^2}-\frac{\partial^2 u}{\partial x^2}-\frac{\partial^2 u}{\partial y^2}+2u=10$$

【求解】 由给定的偏微分方程，可以得出 $c=1, a=2, f=10, d=1$。这样单击偏微分方程界面工具栏中的 PDE 图标，则将打开一个类似于图 7-25 的对话框，左侧有各种常见的偏微分方程类型。选择其中的 Parabolic 选项，就能将给定的偏微分方程的参数输入到该对话框中。

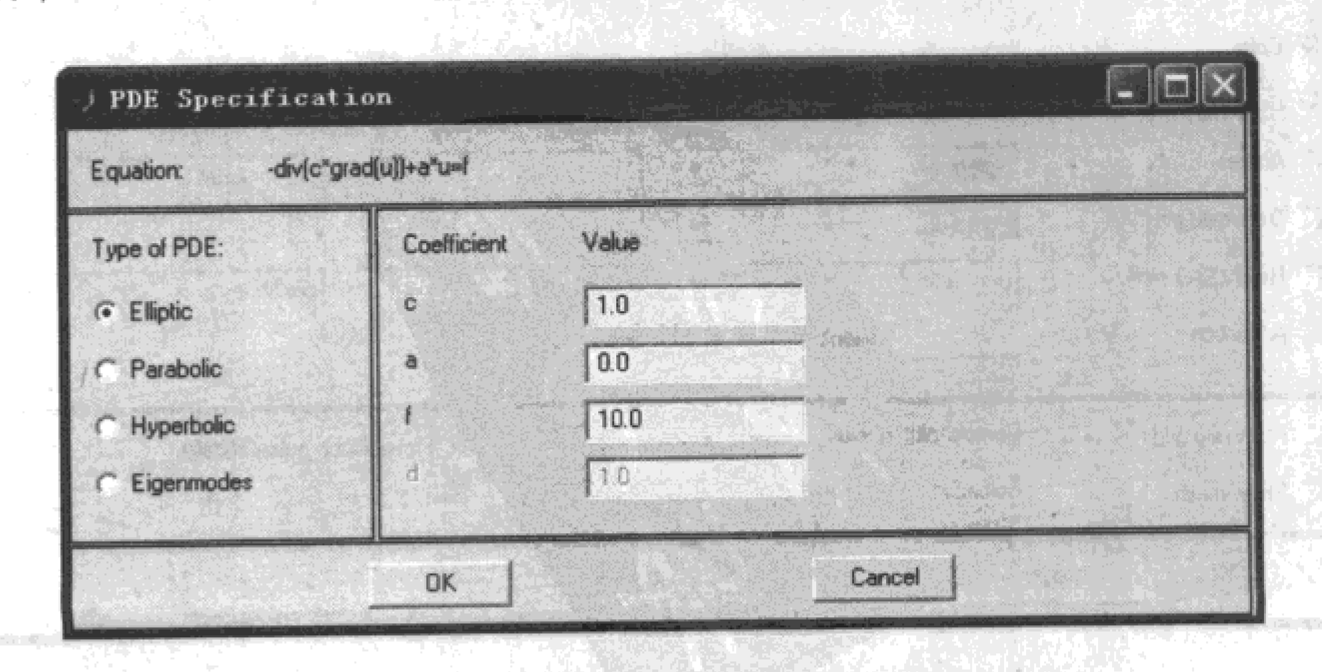
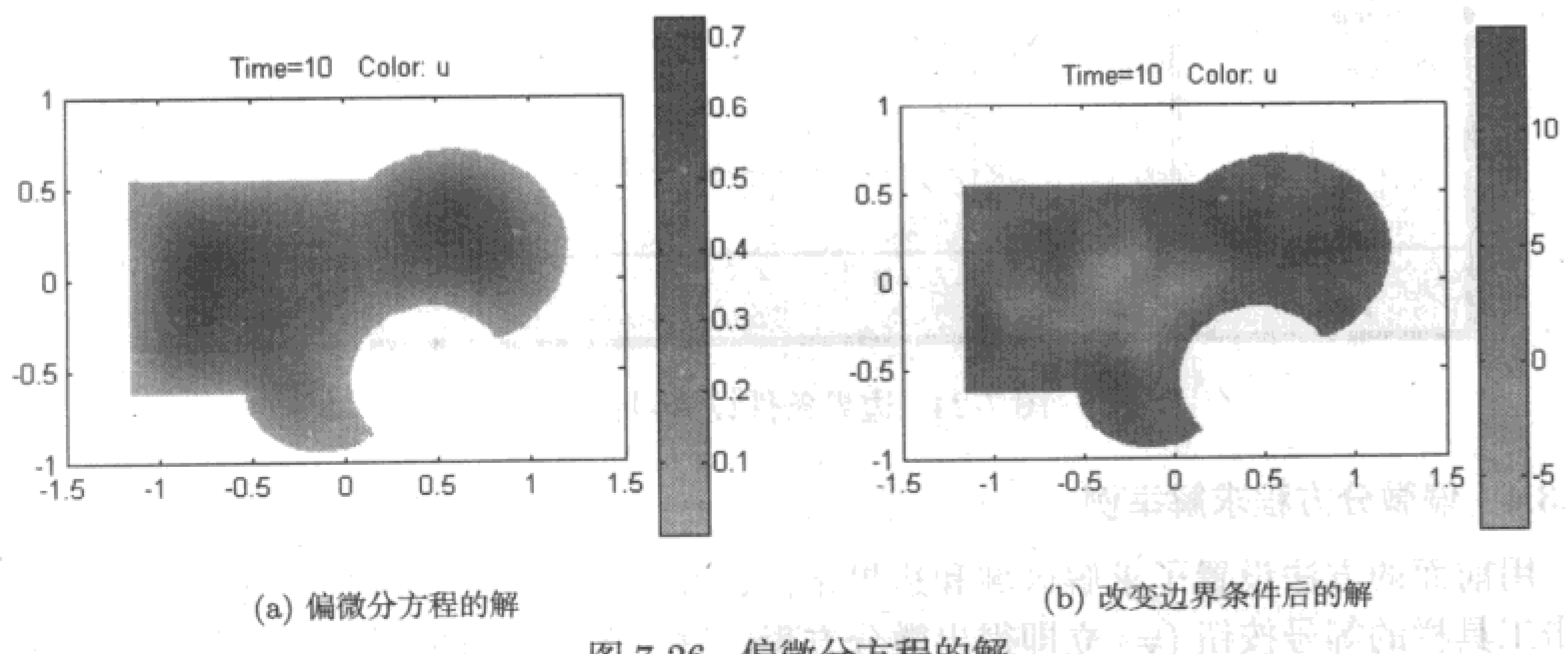


图 7-25 偏微分方程参数设置对话框

如果想求解该偏微分方程，则可以单击工具栏中的等号按钮，这样马上就能求出微分方程的解，如图 7-26 (a) 所示。其中，图形为伪色彩图形，其颜色表示 $u(x,y)$ 值。注意，这时给出的 $u(x,y)$ 的值为在 $t=0$ 时的函数值，后面将介绍如何显示不同 t 下方程的解。

用户还可以修改微分方程的边界条件。例如，再得出图 7-24 所示的对话框，仍采用 Dirichlet 条件，令边界上所有的 u 值为 5，则可以将该对话框中 r 栏目的值填写为 5，这样再求解偏微分方程，将得出如图 7-26 (b) 所示的结果。



(a) 偏微分方程的解　　(b) 改变边界条件后的解

图 7-26 偏微分方程的解

微分方程的结果还可以用其他很多方式显示。单击工具栏中的三维图图标则将打开一个如图 7-27 所示的对话框，若再选择 Contour 则可以绘制等值线图，若选择 Arrows 选项，将计算并绘制引力线，选择这两个选项，将得出如图 7-28 (a) 所示的计算结果。

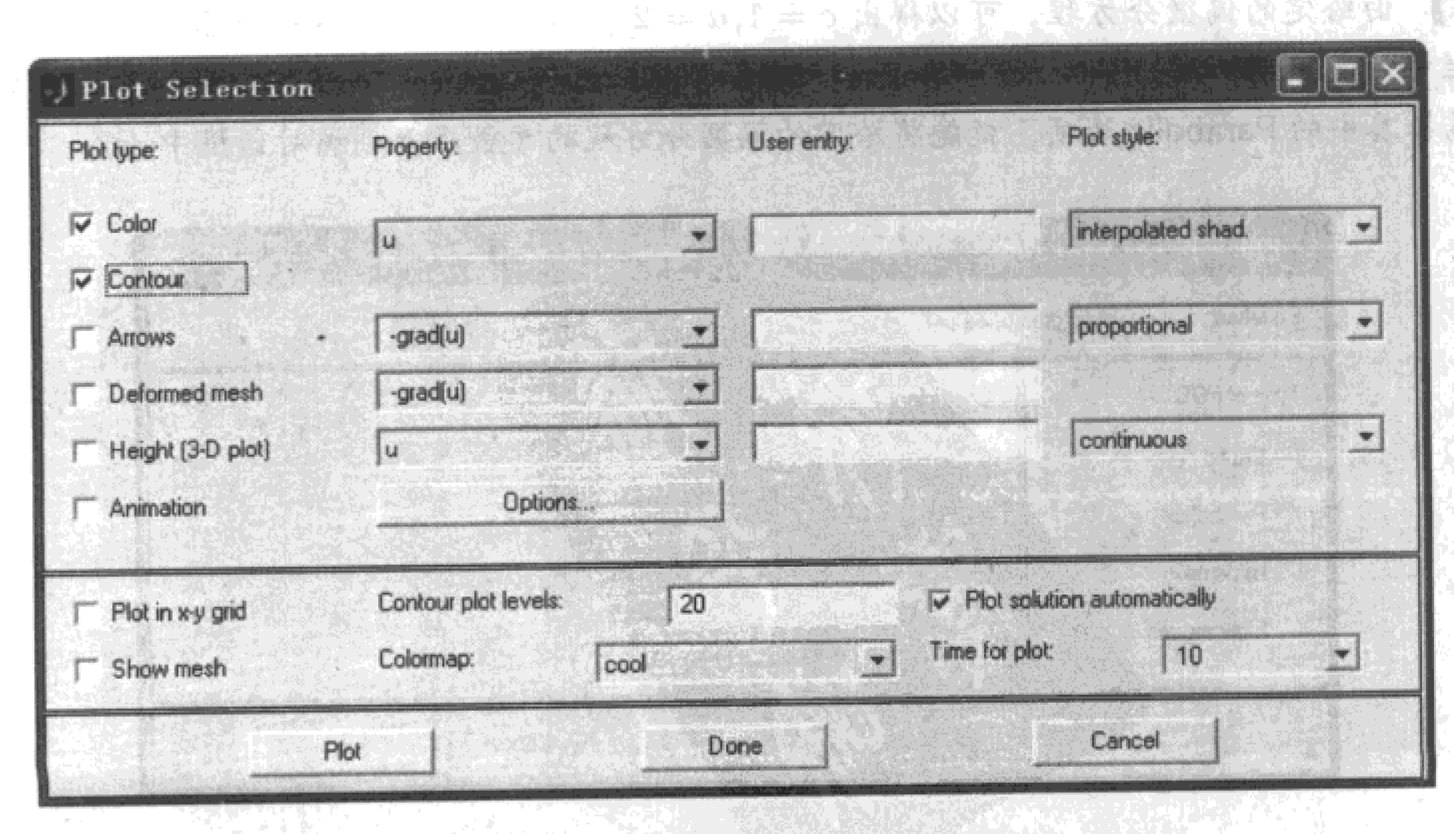
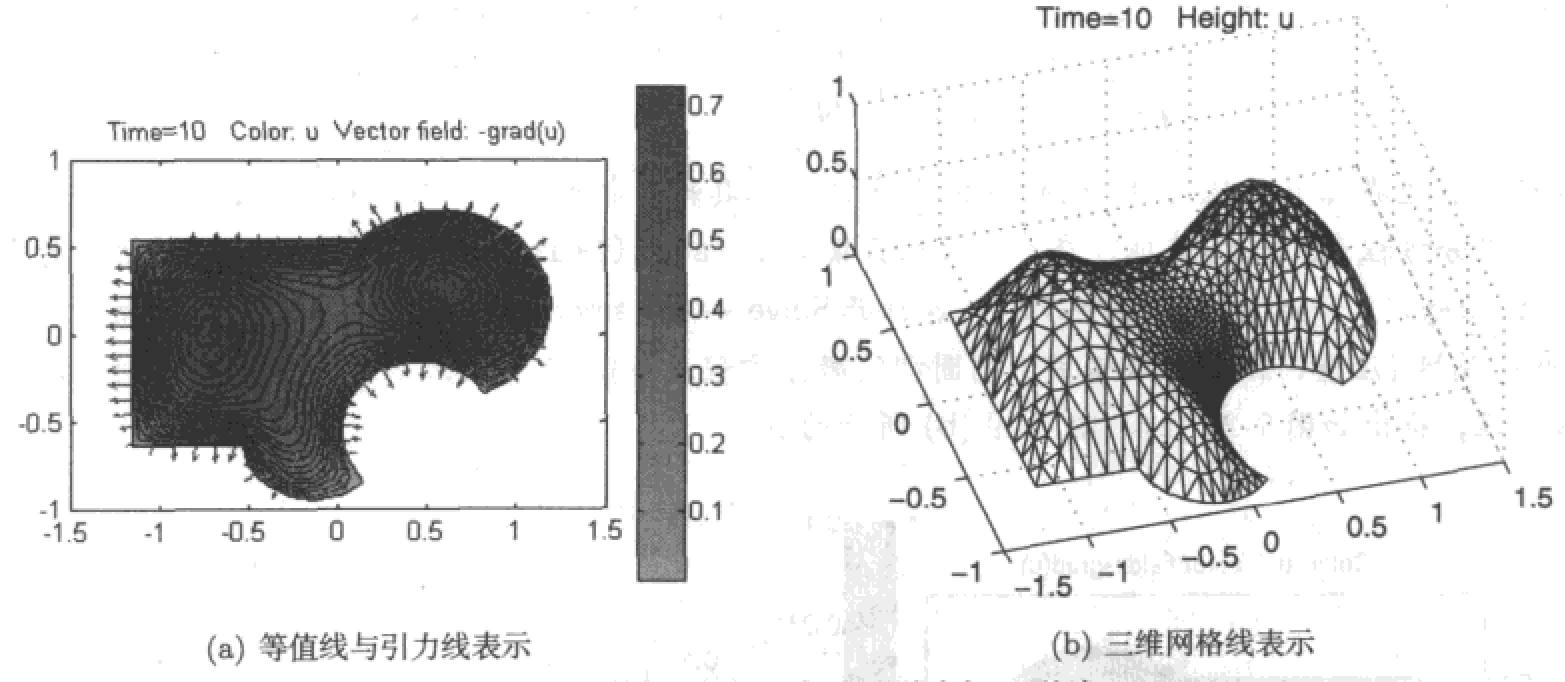


图 7-27 结果显示方式设置对话框

另外还应注意，在如图 7-27 所示的对话框中，Property 栏目的各个项目均有列表框，如第一项的默认值为 u，表明所有的分析都是针对 $u(\cdot)$ 函数的，在绘图时显示的是 $u(x, y)$.如果想显示其他的内容，则可以单击右侧的 ▼，这样就可以打开列表框，从中选择其他的分析内容，直至选择用户自定义栏目。

若单独选择 Height (3d-plot)，则将另外打开一个图形窗口，绘制出网格型三维图形，如图 7-28 (b) 所示。



(a) 等值线与引力线表示 (b) 三维网格线表示

图 7-28 偏微分方程解的不同表现形式

7.5.3.5 时变解的动画显示

偏微分方程程序默认的时间向量为 **t**=0:10，图 7-26 中得出的微分方程解也是在最终时刻 $t = 10$ 的解。从双曲偏微分方程看，方程的解应该是时间 t 的函数，所以应该用动画形式显示出来。现在仍然以例 7-28 中给出的双曲偏微分方程为例，介绍如何将时变的方程显示出来。

用户可以由 Solve → Parameters 菜单引出的对话框设置时间向量，例如在该栏目内填写 `0:0.1:4`，这样再进行微分方程求解则为这段时间的解。定义了该时间向量，由图 7-27 给出的对话框可以选择其中的动画 (Animation) 选项，单击 Options 按钮还可以设置动画的播放速度，如用默认的 6fps (每秒 6 桢)，这样就可以直接获得该微分方程解的动画了。用户可以用 Plot → Export Movie 菜单将动画输出到 MATLAB 工作空间，例如存成变量 `M`，则可以用 `movie(M)` 在 MATLAB 图形窗口中播放得出的动画，也可以用 `movie2avi(M,'myavi.avi')` 命令将动画存成 myavi.avi 文件，以备过后播放。

7.5.3.6 函数参数的偏微分方程求解

前面介绍的偏微分方程 c, a, d, f 均为常数，而在实际遇到的偏微分方程中，经常需要这些量为函数的情况。偏微分方程工具箱目前能处理的问题是含有非线性系数的椭圆偏微分方程问题，在系数字符串中允许使用 x,y 直接表示微分方程中的 x_1, x_2 或 x, y，用变量 ux 和 uy 表示 $\partial u/\partial x$ 和 $\partial u/\partial y$，这样就可以描述任意的非线性系数。下面将通过例子演示这样方程的求解方法。

【例 7-29】假设偏微分方程为

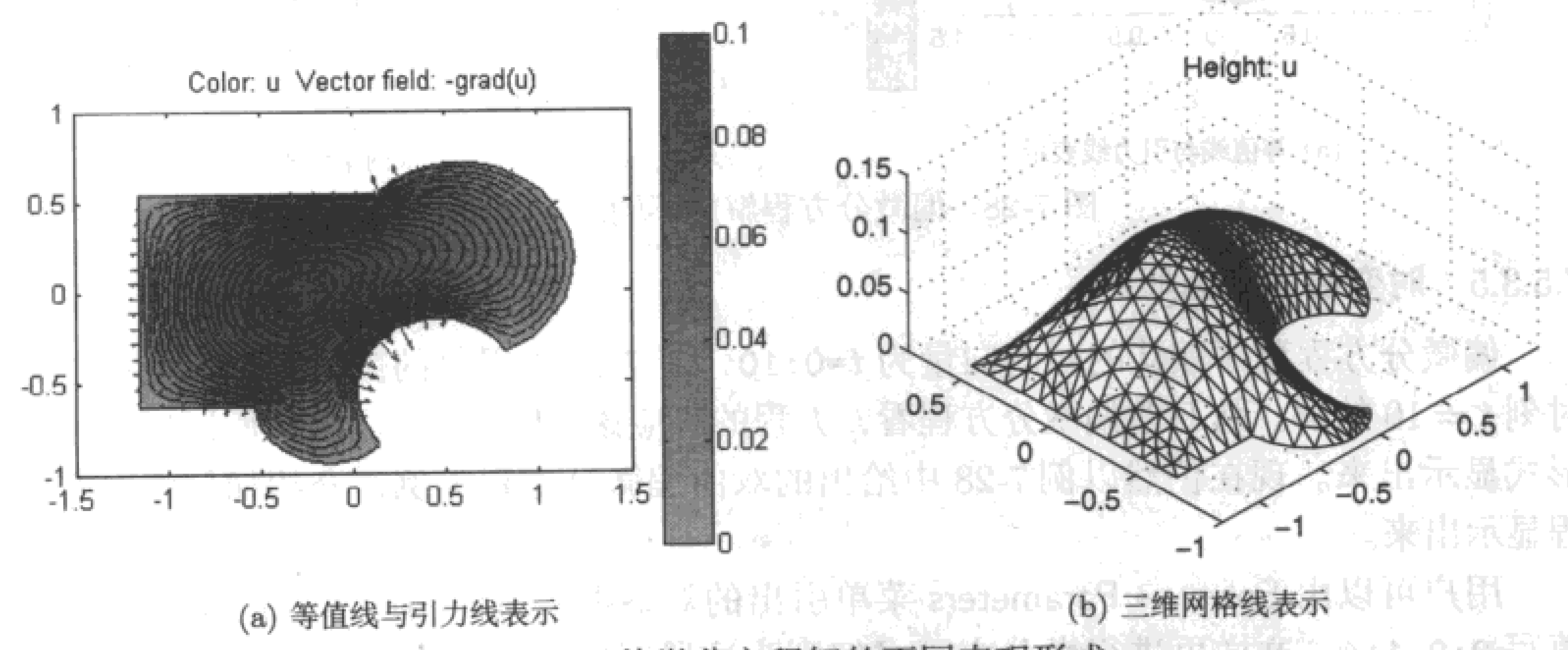
$$-\mathrm{div}\left(\frac{1}{1+|\nabla u|^2}\nabla u\right)+(x^2+y^2)u=\mathrm{e}^{-x^2-y^2}$$

其中边界为 0，试求解该偏微分方程。

【求解】观察该方程可以发现，它满足椭圆型偏微分方程，其中

$$c=\frac{1}{\sqrt{(1+\left(\dfrac{\partial u}{\partial x}\right)^2+\left(\dfrac{\partial u}{\partial y}\right)^2}},\ a=x^2+y^2,\ f=\mathrm{e}^{-x^2-y^2}$$

且在求解边界上 u 的值为 0。仍使用偏微分方程工具箱，打开如图 7-25 所示的对话框，选择椭圆型偏微分方程 (Elliptic) 选项，在 c 参数栏目填写 `1./sqrt(1+ux.^2+uy.^2)`，在 a 和 f 栏目分别填写 `x.^2+y.^2` 和 `exp(-x.^2-y.^2)`，再打开 Solve → Parameters 对话框，从中选定 Use nonlinear solver 属性（注意，该属性只适用于椭圆型偏微分方程求解）。再单击工具栏内的等号，则可以求解该方程，得出如图 7-29 (a)、图 7-29 (b) 所示的解。



(a) 等值线与引力线表示　　(b) 三维网格线表示

图 7-29　偏微分方程解的不同表现形式

7.6　微分方程的框图求解

7.6.1　Simulink 简介

Simulink 环境是 1990 年前后由 The MathWorks 公司推出的产品，原名为 SimuLAB，1992 年改为 Simulink。其名字有两重含义，仿真 (simu) 与模型连接 (link)，表示该环境可以用框图的方式对系统进行仿真。可以利用这一有效的工具，用图形的方式描述各种各样的微分方程，从而求解相应的微分方程。

当然，Simulink的功能远不止微分方程的求解，它还提供了各种可用于控制系统仿真的模块，支持一般的控制系统仿真。此外，它还提供了各种工程应用中可能使用的模块，如电机系统、机构系统、通信系统等的模块集，直接进行建模与仿真研究。Simulink

的功能十分强大，可以借用其本身或模块集对任意复杂的系统进行仿真。相关内容可以参阅其手册[28]和书籍[52]，限于本书篇幅，只能介绍和微分方程求解有关的内容。

本节先简单介绍微分方程建模可能用到的模块，然后通过例子演示微分方程的 Simulink 建模方法与求解方法。

7.6.2 Simulink 相关模块

在 MATLAB 命令窗口下给出下面的命令

```
open_system('simulink')
```

将打开如图 7-30 所示的模块组窗口。可见，该组窗口中提供了各类下一级的模块组，如输入信号源模块组 Sources、连续模块模块组 Continuous 等，每组的模块都是很丰富的，理论上可以建立任意复杂问题的仿真模型。

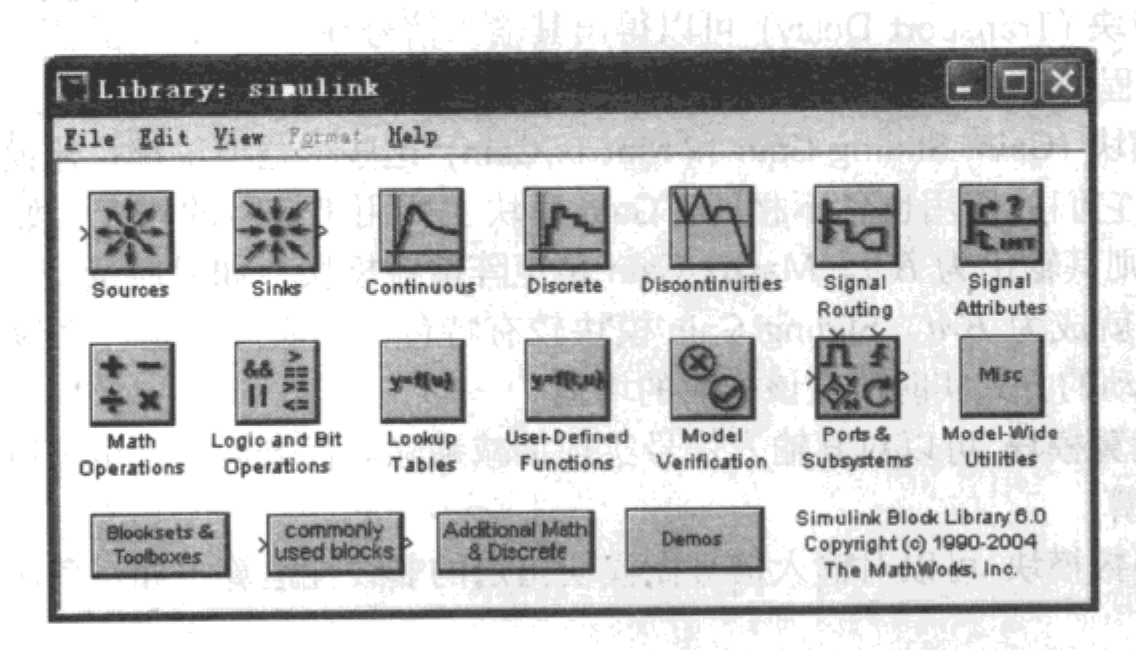


图 7-30 Simulink 环境的主窗口

Simulink 下支持的模块不胜枚举，不可能在本节的篇幅内全部介绍，所以针对微分方程模块搭建问题，作者只选择了一些常用模块作为子模块库，用 odegroup 命令可以打开如图 7-31 所示的自定义模块集。

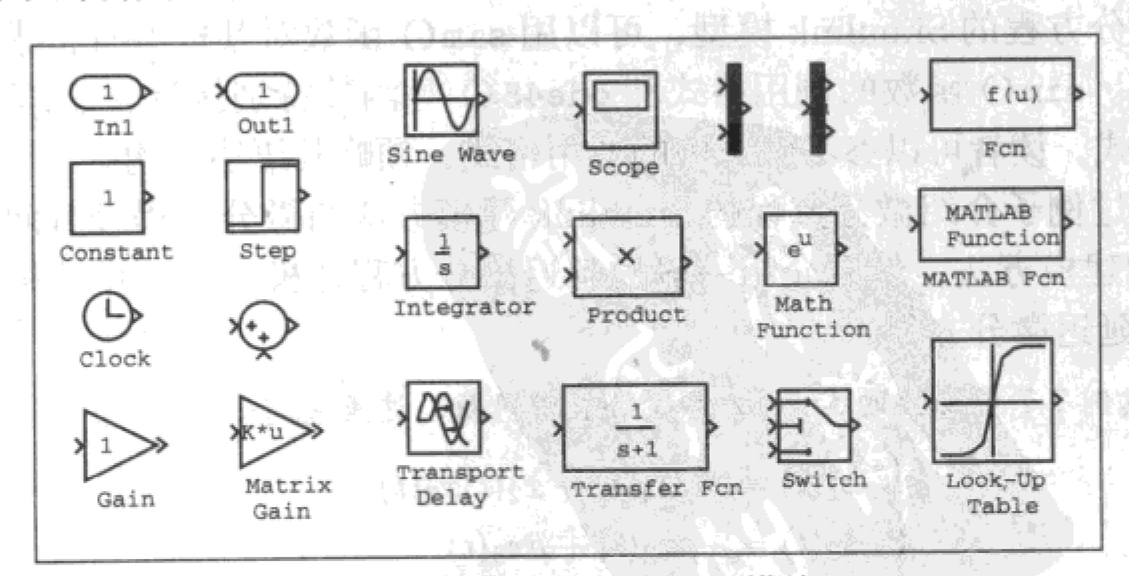


图 7-31 常用模块的自定义模块组

下面将介绍其中常用的模块：

① **输入输出端口** (In1, out1) 一般更常用输出模块显式微分方程求解的结果，该模块将在 MATLAB 工作空间中产生变量 yout。仿真模型中任意一路信号都可以用示波器 Scope 直接显示。

② **时钟模块** Clock 产生时间 t，从而可以搭建时变微分方程模型。

③ **常用输入模块** 可以用 Sine 模块产生正弦信号，用 Step 可以产生阶跃信号，而用 Constant 模块可以产生恒值信号。

④ **积分器模块** (Int) 可以用其描述一阶导数，令常微分方程组的每个一阶导数项作为每个积分器模块的输入。例如，第 i 个积分器模块的输入端定义为 $\dot{x}_i(t)$，则其输出端自然就是 $x_i(t)$。若给出了一阶微分方程组，则积分器输入端的搭建就是整个微分方程组 Simulink 模型搭建的关键。对于线性高阶微分方程，还可以采用其中的传递函数模块 Transfer Function。

⑤ **延迟模块** (Transport Delay) 可以得出其输入信号在 $t-\tau$ 时刻的值。该模块可以用于延迟微分方程的建模与求解。

⑥ **增益模块** (Gain, Sliding Gain 和 Matrix Gain) 这些增益模块都是建模中很有意义的增益模块，而它们的作用也各不相同。Gain 模块主要用于信号的放大，如果该模块的输入信号为 u，则其输出为 Ku。Matrix Gain 是矩阵增益模块，如果向量型输入信号为 $\boldsymbol{u}$，则其输出信号就成为 $\boldsymbol{Ku}$。Sliding Gain 模块较有特色，它实际上是一个滚动杆，用户可以通过鼠标拖动的方式实时改变该模块的增益。

⑦ **数学运算模块** 可以对其输入信号实现加减乘除等代数运算，也可以实现各种逻辑运算和比较运算。

⑧ **数学函数模块** 可以对输入信号做模块指定的非线性运算，如三角函数运算、指数对数运算等。

⑨ **信号向量化模块** 用混路模块 Mux 可以将若干路信号混成向量型的信号，用 DeMux 模块可以将向量型信号解出单路的信号。

7.6.3 微分方程的 Simulink 建模与求解

建立起微分方程的 Simulink 模型，可以用 `sim()` 函数对其模型直接求解，得出微分方程的数值解。`sim()` 函数的调用格式和 `ode45()` 等函数特别接近， 这里不详细介绍该函数的调用格式，读者可以参考例子中的调用格式，领略其语法结构。

本节将通过例子介绍微分方程的 Simulink 建模方法与微分方程求解问题，首先介绍 Lorenz 方程的建模方法；然后介绍一般的延迟微分方程建模，最后用建模的方法解出前面不能求解的延迟微分方程。

【例 7-30】考虑例 7-7 中给出的 Lorenz 方程的求解问题，这里重写如下：

$$\begin{cases}\dot{x}_1(t)=-\beta x_1(t)+x_2(t)x_3(t)\\ \dot{x}_2(t)=-\rho x_2(t)+\rho x_3(t)\\ \dot{x}_3(t)=-x_1(t)x_2(t)+\sigma x_2(t)-x_3(t)\end{cases}$$

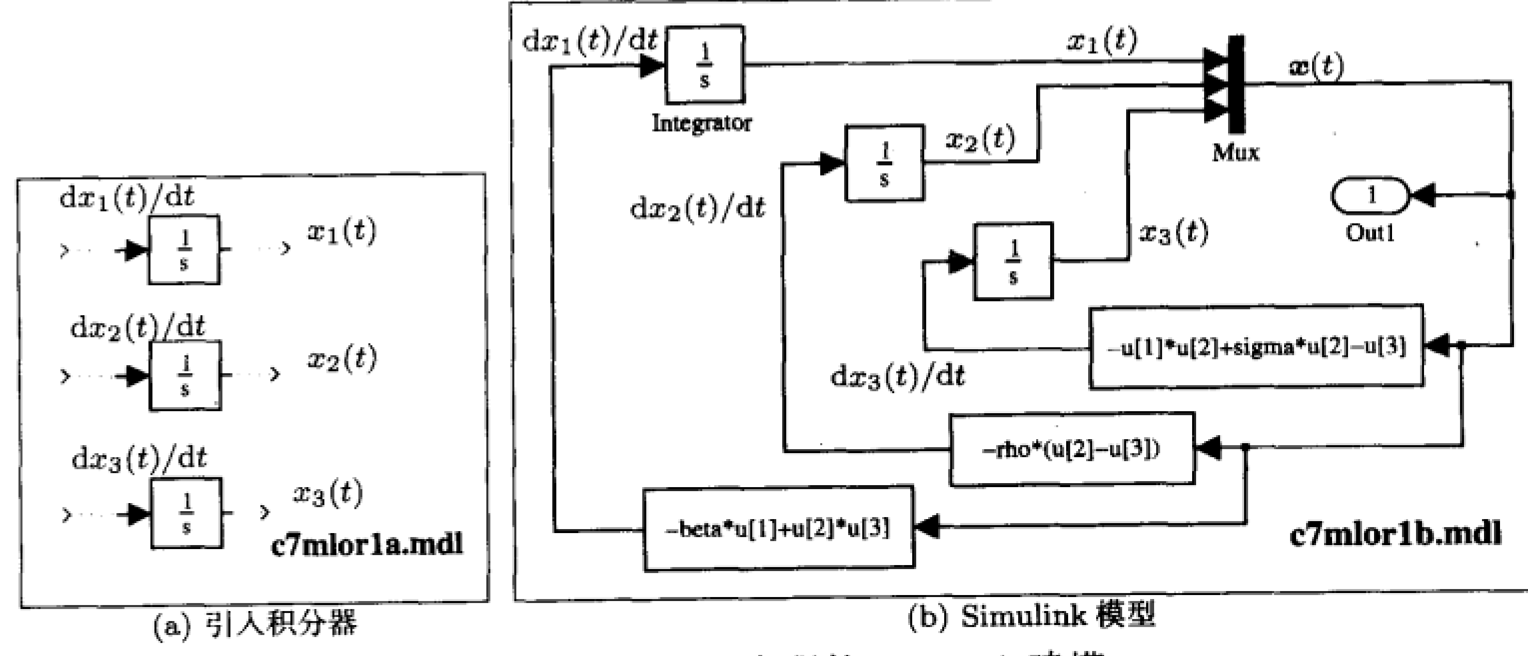
其中，设 $\beta=8/3,\rho=10,\sigma=28$，且各个状态变量的初值为 $x_1(0)=x_2(0)=0$, $x_3(0)=10^{-10}$,

试用 Simulink 搭建该模型，并得出仿真结果。

【求解】 前面已经介绍过，用 MATLAB 语言可以编写一个函数，描述原来的微分方程组模型，然后用 ode45() 类函数就可以直接求解该方程。

用 Simulink 也可以描述出微分方程组的模型。具体的方法是：考虑原方程中有状态变量的一阶导数项，需要使用三个积分器，用其输入端分别描述 $\dot{x}_1(t)$, $\dot{x}_2(t)$, $\dot{x}_3(t)$，则其输出端自然就成了 $x_1(t)$, $x_2(t)$, $x_3(t)$，这样就建立起了 Simulink 框图的核心模块框架，如图 7-32 (a) 所示。双击积分器模块，可以将状态变量初值填写进各个积分器模块。

在构造微分方程求解框架时，定义了各个状态变量及其导数的信号，利用混路器 Mux 模块，可以定义出向量型的信号 $\boldsymbol{x}(t)=[x_1(t),x_2(t),x_3(t)]^{\mathrm{T}}$，该信号进入 Fcn 模块可以直接表示为该模块输入信号的 $\boldsymbol{u}(t)=[u_1(t),u_2(t),u_3(t)]^{\mathrm{T}}$。这样，考虑 Lorenz 方程的第一个式子，可以在最下面的 Fcn 模块中填写 -beta*u[1]+u[2]*u[3]，从而将该模块的输出直接连接到第一个积分器的输入端 $\mathrm{d}x_1/\mathrm{d}t$，搭建起第一个微分方程式。用类似的方法可以建立起其他两个微分方程式，用这样的方法最终可以构造出如图 7-32 (b) 所示的完整 Simulink 模型。为获得仿真结果，可以将 $\boldsymbol{x}(t)$ 状态变量连接到输出端口。



(a) 引入积分器 (b) Simulink 模型

图 7-32 Lorenz 方程的 Simulink 建模

仿真模型建立起来之后，可以用下面的语句对该微分方程进行求解，并得出和图 7-3 (a)、图 7-3 (b) 完全一致的仿真结果，这里不再重复给出。

```
>> beta=8/3; rho=10; sigma=28;
   [t,x]=sim('c7mlor1b',[0,100]); plot(t,x)
   figure; plot3(x(:,1),x(:,2),x(:,3))
```

注意，这里的 beta, rho 和 sigma 参数可以写入 MATLAB 工作空间，而无需作为附加参数在语句调用中给出，也可以将积分器的初值在积分器模块中设置成变量形式，无需改变 Simulink 模型本身就可以改变状态变量的初值。

应该指出，对于小规模问题来说，用 Simulink 建模的求解方式比用 ode45() 等函数的调用方式更复杂。但在解决大规模问题、模块化问题亦即复杂混连系统的问题时，用 Simulink 建模方式应该比简单的函数调用更合适。此外，对更复杂的时间延迟微分方程求解问题来说，采用 Simulink 建模的方法，可以解决用普通微分方程求解函数解决不了的问题。

【例 7-31】 考虑例 7-22 中介绍的延迟微分方程式，重写如下

$$\begin{cases} \dot{x}(t) = 1 - 3x(t) - y(t-1) - 0.2x^3(t-0.5) - x(t-0.5) \\ \ddot{y}(t) + 3\dot{y}(t) + 2y(t) = 4x(t) \end{cases}$$

试用 Simulink 搭建该微分方程模型，并得出其数值解。

【求解】 该模型当然可以用 MATLAB 语言编写一个函数来表示，然后用延迟微分方程的求解函数 dde23() 直接求解，但这样的求解过程似乎不是很直观。

现在考虑第一个方程式，将 $-3x(t)$ 项移动到等号左侧，则可以将其变换成

$$\dot{x}(t) + 3x(t) = 1 - y(t-1) - 0.2x^3(t-0.5) - x(t-0.5)$$

所以可以将该方程理解成 $x(t)$ 为传递函数模型 $1/(s+3)$ 的输出信号，而该函数的输入信号为 $1 - y(t-1) - 0.2x^3(t-0.5) - x(t-0.5)$。第 2 个方程式可以理解为：$y(t)$ 是传递函数模块 $4/(s^2+3s+2)$ 的输出信号，而该模块的输入信号为 $x(t)$，在 $x(t)$ 信号和 $y(t)$ 信号上连接延迟模块 Transport Delay 可以得出这些信号的延迟。通过说明的分析，可以搭建出如图 7-33 所示的 Simulink 仿真模型。

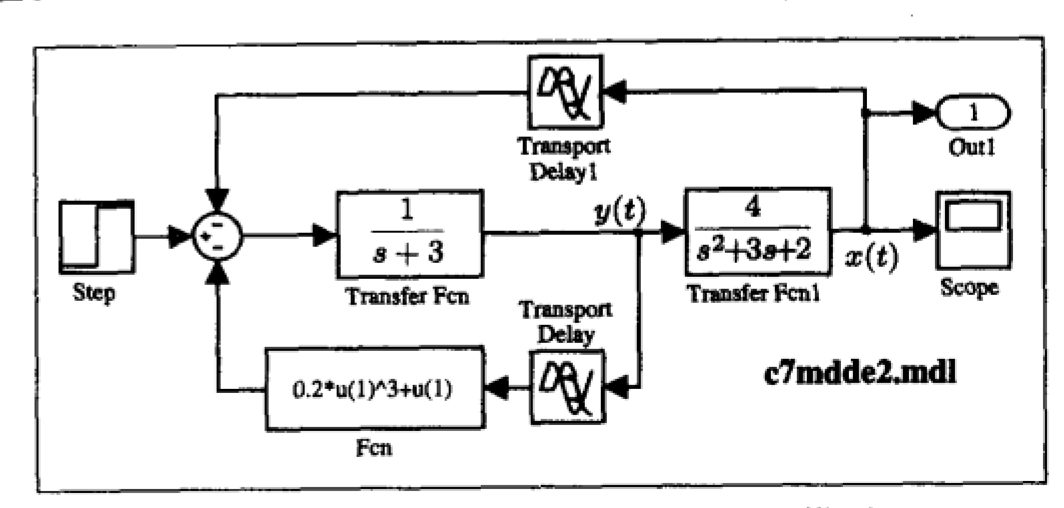


图 7-33 延迟微分方程的 Simulink 模型

建立了仿真模型后，就可以用下面的语句求解该微分方程，并得出输出信号 $y(t)$ 的曲线。该结果和例 7-22 中给出的完全一致，这里不再重复给出。该结果还可以同时在示波器上显示出来。

```
>> [t,x]=sim('c7mdde2',[0,10]); plot(t,x)
```

当然，若不习惯使用传递函数模块，还可以假设 $x_1 = x, x_2 = y, x_3 = \dot{y}$，这样可以将原微分方程模型变换成一阶状态方程模型

$$\begin{cases} \dot{x}_1(t) = 1 - x_1(t) - x_2(t-1) + 0.2x_1^3(t-0.5) - x_1(t-0.5) \\ \dot{x}_2(t) = x_3(t) \\ \dot{x}_3(t) = -4x_1(t) - 3x_3(t) - 2x_2(t) \end{cases}$$

给这 3 个状态变量选择 3 个积分器，则可以搭建出 Simulink 框图，也可以得出同样的结果。这里不给出具体的 Simulink 模型，读者可以按习题的要求自己搭建该模型。

【例 7-32】 现在考虑例 7-23 中定义的延迟微分方程，其中

$$\boldsymbol{A}_1 = \begin{bmatrix} -13 & 3 & -3 \\ 106 & -116 & 62 \\ 207 & -207 & 113 \end{bmatrix},\ \boldsymbol{A}_2 = \begin{bmatrix} 0.02 & 0 & 0 \\ 0 & 0.03 & 0 \\ 0 & 0 & 0.04 \end{bmatrix},\ \boldsymbol{B} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

试用 Simulink 搭建系统模型，并得出系统的仿真曲线。

【求解】该方程用 MATLAB 自身提供的 dde23() 函数无法求解，所以这里考虑采用基于 Simulink 的框图形式求解该方程。在建模之前，可以用下面的语句输入已知的矩阵：

```
>> A1=[-13,3,-3; 106,-116,62; 207,-207,113];
   A2=diag([0.02,0.03,0.04]); B=[0; 1; 2];
```

再考虑原始的微分方程模型，已经存在一个状态向量 $x(t)$，故可以安排一个积分器，使得其输出为 $x(t)$，这样其输入端自然是 $\dot{x}(t)$，可以分别给这两个信号连接延迟环节，并按实际情况设置延迟时间常数，则可以构造出 $x(t-\tau_1)$ 和 $\dot{x}(t-\tau_2)$ 信号，这样经过简单的处理就可以搭建出如图 7-34 所示的 Simulink 模型。

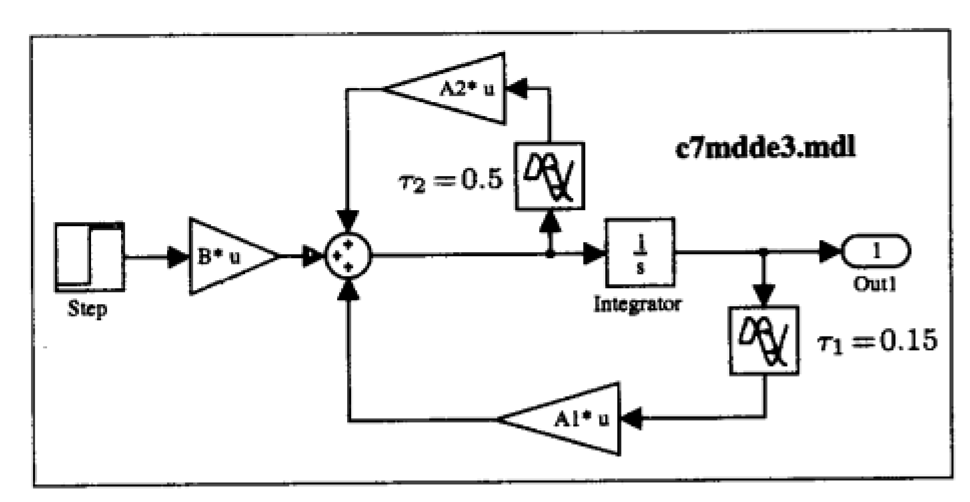


图 7-34 带有导数延迟的微分方程 Simulink 模型

用下面的语句就可以求解该方程，并将各个状态变量绘制出来，如图 7-35 所示。

```
>> [t,x]=sim('c7mdde3',[0,8]); plot(t,x)
```

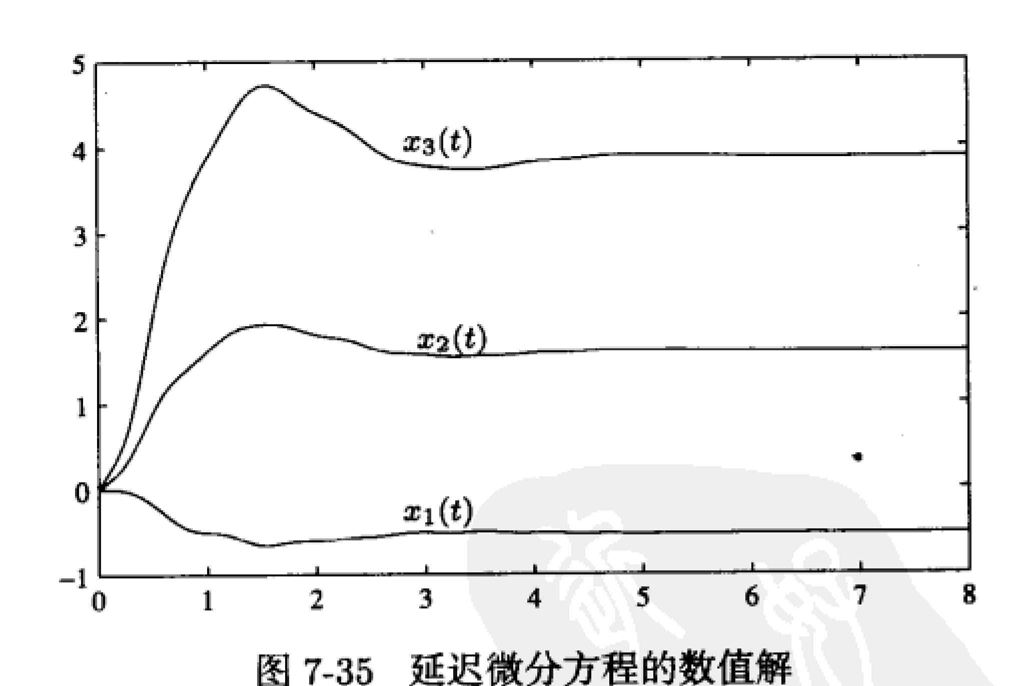


图 7-35 延迟微分方程的数值解

7.7 本章要点简介

- 本章有关的 MATLAB 函数及自编函数由下表给出。

函数名	函 数 功 能	工具箱	本书页码
dsolve()	常微分方程的解析解，尤其适用于线性常微分方程组	符号运算	204
语句	利用 Laplace 变换求解微分方程	自编	207
rk_4()	用定步长四阶 Runge-Kutta 法求解常微分方程组	自编	212
ode45()	用四阶五级 Runge-Kutta-Fahlberg 变步长算法求解常微分方程组，类似的函数还有 ode23()，ode15s()，ode113()，ode23s()，ode23t，ode23tb() 等，适用于一般微分方程、刚性微分方程、微分代数方程、隐式微分方程等的直接求解	MATLAB	213
odeset()	微分方程的控制参数	MATLAB	214
dde23()	延迟微分方程的数值求解	MATLAB	233
shooting()	线性微分方程的边值问题的打靶算法	自编	235
nlbound()	非线性微分方程的边值问题的打靶算法	自编	238
fdiff()	有限差分算法求解线性微分方程的边值问题	自编	239
pdepe()	偏微分方程的数值解	MATLAB	241
pdetool()	偏微分方程求解界面	偏微分方程	244
open_system()	启动 Simulink 环境或模型	Simulink	251
odegroup	常微分方程组建模常用模块库	自编	251
sim()	Simulink 模型的仿真求解	Simulink	252

- 本章介绍了基于 MATLAB 符号运算工具箱 dsolve() 函数的线性微分方程的解析解方法，并介绍基于该函数的特殊非线性微分方程的解析解。然而对一般非线性微分方程来说，解析解是不存在的，只能依赖数值解的方法对其进行研究。
- 引入了数值解的概念，并以最简单的一阶微分方程的 Euler 算法为例，介绍了一般数值解法的思路并介绍了变步长求解的概念，还介绍了 MATLAB 下的微分方程数值求解函数 ode45()，通过例子演示了该函数的使用方法。
- 微分方程初值函数能直接求解的方程是一阶显式微分方程组，若给出的方程不是这类函数，则需要通过本书介绍的方法选择一组状态变量，将原方程变换成一阶显式微分方程组，以便用给定的求解函数直接求解。
- 若某微分方程模型求解速度极慢，则有可能为刚性方程，需要调用 ode15s() 等函数来求解。此外，其他类型的微分方程，如微分代数方程、隐式微分方程与延迟微分方程等，也可以由 MATLAB 语言提供的现成函数直接求解。
- 二阶微分方程的边值问题可以用本书提供的 3 种算法求解。
- 偏微分方程可以由 MATLAB 提供的现成函数直接求解，x-y 平面的偏微分方程可以由 MATLAB 语言的偏微分方程工具箱提供的界面直接求解，而高维偏微分方程可以由该工具箱提供的现成函数直接求解。
- Simulink 是 MATLAB 中的一个很重要的系统仿真平台，可以用该高阶以框图的形式建立起系统的模型。本书先介绍其入门知识，然后侧重于微分方程求解，介绍了

Simulink 如何搭建微分方程框图，其中一个重要的方法就是用积分器来定义状态变量和其导数，用已知信号搭建起这样的微分方程，然后用该工具提供的求解按钮直接求解。

7.8 习 题

1 试求出下面线性微分方程的通解。

$$\frac{\mathrm{d}^5y(t)}{\mathrm{d}t^5}+13\frac{\mathrm{d}^4y(t)}{\mathrm{d}t^4}+64\frac{\mathrm{d}^3y(t)}{\mathrm{d}t^3}+152\frac{\mathrm{d}^2y(t)}{\mathrm{d}t^2}+176\frac{\mathrm{d}y(t)}{\mathrm{d}t}+80y(t)=\mathrm{e}^{-2t}\left[\sin\left(2t+\frac{\pi}{3}\right)+\cos(3t)\right]$$

假设上述微分方程满足已知条件 $y(0)=1, y(1)=3, y(\pi)=2, \dot{y}(0)=1, \dot{y}(1)=2$，试求出满足该条件的微分方程的解析解。

2 试求解下面微分方程的通解以及满足 $x(0)=1, x(\pi)=2, y(0)=0$ 条件下的解析解。

$$\begin{cases} \ddot{x}(t)+5\dot{x}(t)+4x(t)+3y(t)=\mathrm{e}^{-6t}\sin(4t) \\ 2\dot{y}(t)+y(t)+4\dot{x}(t)+6x(t)=\mathrm{e}^{-6t}\cos(4t) \end{cases}$$

3 试求出下面的时变线性微分方程解析解。

① Legendre 微分方程 $(1-t^2)\dfrac{\mathrm{d}^2x}{\mathrm{d}t^2}-2t\dfrac{\mathrm{d}x}{\mathrm{d}t}+n(n+1)x=0$

② Bessel 微分方程 $t^2\dfrac{\mathrm{d}^2x}{\mathrm{d}t^2}+t\dfrac{\mathrm{d}x}{\mathrm{d}t}+(t^2-n^2)x=0$

4 试求出微分方程 $\ddot{y}(x)-\left(2-\dfrac{1}{x}\right)\dot{y}(x)+\left(1-\dfrac{1}{x}\right)y(x)=x^2\mathrm{e}^{-5x}$ 的解析解通解，并求出满足边界条件 $y(1)=\pi$, $y(\pi)=1$ 的解析解。

5 试用 Laplace 变换求解下面微分方程组的解析解，并和其他方法比较。

$$\begin{cases} \ddot{x}(t)+\ddot{y}(t)+x(t)+y(t)=0,\ x(0)=2, y(0)=1 \\ 2\ddot{x}(t)-\ddot{y}(t)-x(t)+y(t)=\sin t,\ \dot{x}(0)=\dot{y}(0)=-1 \end{cases}$$

6 试求出下面微分方程的通解。

① $\ddot{x}(t)+2t\dot{x}(t)+t^2x(t)=t+1$ ② $\dot{y}(x)+2xy(x)=x\mathrm{e}^{-x^2}$ ③ $y^{(3)}+3\ddot{y}+3\dot{y}+y=\mathrm{e}^{-t}\sin t$

7 极限环是二阶非线性常微分方程中一种常见的现象，对某些非线性微分方程来说，不论初始状态为何值，微分方程的相轨迹都将稳定在一条封闭的曲线上，该曲线称为微分方程的极限环。试绘制出微分方程 $\begin{cases} \dot{x}=y+x(1-x^2-y^2) \\ \dot{y}=-x+y(1-x^2-y^2) \end{cases}$ 的极限环，并对不同初值验证微分方程的相平面曲线确实收敛于极限环。

8 考虑著名的 Rössler 化学反应方程组 $\begin{cases} \dot{x}=-y-z \\ \dot{y}=x+ay \\ \dot{z}=b+(x-c)z \end{cases}$，选定 $a=b=0.2$, $c=5.7$，且 $x_1(0)=x_2(0)=x_3(0)$，绘制仿真结果的三维相轨迹，并得出其在 x-y 平面上的投影。在实际

求解中建议将 a,b,c 作为附加参数，同样的方程若设 $a=0.2,b=0.5,c=10$ 时，绘制出状态变量的二维图和三维图。

9 Chua 电路方程是混沌理论中经常提到的微分方程[54] $\begin{cases}\dot{x}=\alpha[y-x-f(x)]\\ \dot{y}=x-y+z\\ \dot{z}=-\beta y-\gamma z\end{cases}$，其中，$f(x)$ 为 Chua 电路的二极管分段线性特性，$f(x)=bx+\dfrac{1}{2}(a-b)(|x+1|-|x-1|)$，且 $a<b<0$。试编写出 MATLAB 函数描述该微分方程，并绘制出 $\alpha=15,\beta=20,\gamma=0.5,a=-120/7,b=-75/7$，且初始条件为 $x(0)=-2.121304,y(0)=-0.066170,z(0)=2.881090$ 时的相空间曲线。

10 Lotka-Volterra 扑食模型方程为 $\begin{cases}\dot{x}(t)=4x(t)-2x(t)y(t)\\ \dot{y}(t)=x(t)y(t)-3y(t)\end{cases}$，且初值为 $x(0)=2,y(0)=3$，试求解该微分方程，并绘制相应的曲线。

11 请给出求解下面微分方程的 MATLAB 命令，

$$y^{(3)}+ty\ddot{y}+t^2\dot{y}y^2=\mathrm{e}^{-ty},\ y(0)=2,\ \dot{y}(0)=\ddot{y}(0)=0$$

并绘制出 $y(t)$ 曲线。试问该方程存在解析解吗？选择四阶定步长 Runge-Kutta 算法求解该方程时，步长选择多少可以得出较好的精度，试与 MATLAB 语言给出的现成函数在速度、精度上进行比较。

12 试选择状态变量，将下面的非线性微分方程组转换成一阶显式微分方程组，并用 MATLAB 对其求解，绘制出解的相平面或相空间曲线。

① $\begin{cases}\ddot{x}=-x-y-(3\dot{x})^2+(\dot{y})^3+6\ddot{y}+2t\\ y^{(3)}=-\ddot{y}-\dot{x}-\mathrm{e}^{-x}-t\\ x(1)=2,\dot{x}(1)=-4\\ y(1)=-2,\dot{y}(1)=7,\ddot{y}(1)=6\end{cases}$　　② $\begin{cases}\ddot{x}-2xz\dot{x}=3x^2yt^2\\ \ddot{y}-\mathrm{e}^y\dot{y}=4xt^2z\\ \ddot{z}-2t\dot{z}=2t\mathrm{e}^{xy}\\ \dot{z}(1)=\dot{x}(1)=\dot{y}(1)=2\\ z(1)=x(1)=y(1)=3\end{cases}$

13 试用解析解和数值解的方法求解下面的微分方程组。

$$\begin{cases}\ddot{x}(t)=-2x(t)-3\dot{x}(t)+\mathrm{e}^{-5t}, & x(0)=1,\dot{x}(0)=2\\ \ddot{y}(t)=2x(t)-3y(t)-4\dot{x}(t)-4\dot{y}(t)-\sin t, & y(0)=3,\dot{y}(0)=4\end{cases}$$

14 给定微分方程组 $\begin{cases}\ddot{u}(t)=-u(t)/r^3(t)\\ \ddot{v}(t)=-v(t)/r^3(t)\end{cases}$，其中，$r(t)=\sqrt{u^2(t)+v^2(t)}$，且 $u(0)=1,\dot{u}(0)=2,\dot{v}(0)=2,v(0)=1$，试选择一组状态变量，将其变换成 MATLAB 语言能直接求解的微分方程组形式，并绘制出 $u(t),v(t)$ 的轨迹曲线。

15 已知微分方程可以表示为[30] $\begin{cases}\dot{u}_1=u_3\\ \dot{u}_2=u_4\\ 2\dot{u}_3+\cos(u_1-u_2)\dot{u}_4=-g\sin u_1-\sin(u_1-u_2)u_4^2\\ \cos(u_1-u_2)\dot{u}_3+\dot{u}_4=-g\sin u_2+\sin(u_1-u_2)u_3^2\end{cases}$，其中，

$u_1(0)=45, u_2(0)=30, u_3(0)=u_4(0)=0$，g= 9.81，试求解此微分方程，并绘制出各个状态变量的时间曲线。

16 试求出隐式微分方程 $\begin{cases}\dot{x}_1\ddot{x}_2\sin(x_1x_2)+5\ddot{x}_1\dot{x}_2\cos(x_1^2)+t^2x_1x_2^2=\mathrm{e}^{-x_2^2}\\ \ddot{x}_1x_2+\ddot{x}_2\dot{x}_1\sin(x_1^2)+\cos(\ddot{x}_2x_2)=\sin t\end{cases}$ 的数值解，$x_1(0)=1, \dot{x}_1(0)=1, x_2(0)=2, \dot{x}_2(0)=2$，并绘制出轨迹曲线。

17 下面的方程在传统微分方程教程中经常被认为是刚性微分方程。试用常规微分方程解法和刚性微分方程解法分别求解这两个微分方程的数值解，并求出解析解，用状态变量曲线比较数值求解的精度。

① $\begin{cases}\dot{y}_1=9y_1+24y_2+5\cos t-\dfrac{1}{3}\sin t,\ y_1(0)=\dfrac{1}{3}\\ \dot{y}_2=-24y_1-51y_2-9\cos t+\dfrac{1}{3}\sin t,\ y_2(0)=\dfrac{2}{3}\end{cases}$ ② $\begin{cases}\dot{y}_1=-0.1y_1-49.9y_2,\ y_1(0)=1\\ \dot{y}_2=-50y_2,\ y_2(0)=2\\ \dot{y}_3=70y_2-120y_3,\ y_3(0)=1\end{cases}$

18 考虑下面的化学反应系统的反应速度方程组 $\begin{cases}\dot{y}_1=-0.04y_1+10^4y_2y_3\\ \dot{y}_2=0.04y_1-10^4y_2y_3-3\times10^7y_2^2\\ \dot{y}_3=3\times10^7y_2^2\end{cases}$，其初值为 $y_1(0)=1, y_2(0)=y_3(0)=0$，该方程往往被认为是刚性方程。试采用 `ode45()` 对之求解，观察是否能正确求解，如果不能求解应该如何解决问题?

19 试求出习题 4 中给出的微分方程边值问题数值解，绘制出 $y(t)$ 曲线，并和该习题得出的解析解比较精度。

20 考虑 Van der Pol 方程 $\ddot{y}+\mu(y^2-1)\dot{y}+y=0$，试求解 $\mu=1$，且边值 $y(0)=1, y(5)=3$ 时方程的数值解。如果假设 μ 为自由参数，试求出满足边值条件，且满足 $\dot{y}(5)=-2$ 时方程的数值解及 μ 的值，并绘图验证之。

21 试用数值方法求解偏微分方程 $\begin{cases}\dfrac{\partial^2u}{\partial x^2}+\dfrac{\partial^2u}{\partial y^2}=0\\ u\big|_{x=0,y>0}=1,\ u\big|_{y=0,\ x\geqslant0}=0\\ x>0,\ y>0\end{cases}$，并绘制出 u 函数曲面。

22 考虑简单的线性微分方程 $y^{(4)}+3y^{(3)}+3\ddot{y}+4\dot{y}+5y=\mathrm{e}^{-3t}+\mathrm{e}^{-5t}\sin(4t+\pi/3)$，且方程的初值为 $y(0)=1, \dot{y}(0)=\ddot{y}(0)=1/2, y^{(3)}(0)=0.2$，试用 Simulink 搭建起系统的仿真模型，并绘制出仿真结果曲线。考虑上面的模型，假设给定的微分方程变化成时变线性微分方程 $y^{(4)}+3ty^{(3)}+3t^2\ddot{y}+4\dot{y}+5y=\mathrm{e}^{-3t}+\mathrm{e}^{-5t}\sin(4t+\pi/3)$，而方程的初值仍为 $y(0)=1$，$\dot{y}(0)=\ddot{y}(0)=1/2, y^{(3)}(0)=0.2$，试用 Simulink 搭建起系统的仿真模型，并绘制出仿真结果曲线。

23 考虑延迟微分方程 $y^{(4)}(t)+4y^{(3)}(t-0.2)+6\ddot{y}(t-0.1)+6\ddot{y}(t)+4\dot{y}(t-0.2)+y(t-0.5)=\mathrm{e}^{-t^2}$，且在 $t\leqslant0$ 时该方程具有零初始条件，试分别用 Simulink 建模与 `dde23()` 函数求解的方式直接求解该微分方程，并绘制出 $y(t)$ 曲线。

第 8 章　数据插值、函数逼近问题的计算机求解

在科学与工程研究中经常会通过实验测出一些数据，根据这些数据对某种规律进行研究是数据插值与函数逼近所要解决的问题。可以将已知数据看成是样本点，所谓数据插值就是在样本点的基础上求出不在样本点上的其他点处的函数值。第 8.1 节将介绍一维、二维甚至多维数据插值问题的求解方法，并介绍一种基于插值技术的求取数值积分的方法。第 8.2 节将介绍两种常用的样条插值方式，三次分段多项式的插值方式和 B 样条插值方式，通过例子比较二者的不同，并介绍基于样条插值的数值微积分运算，还将演示该积分运算的结果优于第 8.1 节介绍的方法。掌握了这两节就能较好的求解一维或多维数据的插值运算。

所谓函数逼近问题即由已知的样本点数据求取能对其有较好拟合效果的函数表达式的方法。最简单地，可以由多项式拟合更多的样本点，这样求解使得拟合误差极小化的多项式的系数即为多项式拟合或逼近所要解决的问题，而由有理函数拟合多项式的 Padé 近似及 MATLAB 求解、给定函数原型的函数参数最小二乘拟合等都是很有效的函数逼近方法，在第 8.3 节中将详细介绍这些函数逼近问题。第 8.4 节还将介绍信号的相关分析、给定数据的快速 Fourier 变换技术、噪声滤波技术及滤波器设计等有关的信号处理入门知识及其 MATLAB 语言实现。

本章涉及的内容很多也可以由其他的非传统方法求解，如数据插值、拟合等内容将在第 10.2 节中介绍用神经网络进行研究，而噪声滤波等内容将在第 10.4 节中用小波变换的方式求解，有兴趣的读者可以自己阅读相关内容，并比较这些方法与本章介绍方法之间的优劣。

8.1　插值与数据拟合

8.1.1　一维数据的插值问题

8.1.1.1　一维插值问题的求解

假设 $f(x)$ 是一维给定函数，函数本身未知，只已知在相异的一组 N 个自变量 $x_1, x_2, \cdots, x_N$ 点处的函数值为 $y_1, y_2, \cdots, y_N$，这样一组样本点 (x_1, y_1), (x_2, y_2), $\cdots$, (x_N, y_N) 又经常称为样本点，则由这些已知样本点的信息获得该函数在其他点上函数值的方法称为函数的插值。如果在这些给定点的范围内进行插值，又称为内插，否则称为外插。如果从时间概念上理解这个问题，则对 x_N 以后点的插值又称为预报。

MATLAB 语言中提供了若干个插值函数，如一维插值函数 `interp1()`，多项式拟合函数 `polyfit()` 等，还有大量的解决多维插值问题的函数。

一维插值 `interp1()` 函数的调用格式为

y_1=interp1(x,y,x_1,方法)

其中，$\boldsymbol{x} = [x_1, x_2, \cdots, x_N]^{\mathrm{T}}$，$\boldsymbol{y} = [y_1, y_2, \cdots, y_N]^{\mathrm{T}}$，两个向量分别表示给定的一组自变量和函数值数据，可以用这这两个向量来表示已知的样本点坐标，且不要求 $\boldsymbol{x}$ 向量为单调的。$\boldsymbol{x}_1$ 为用户指定的一组新的插值点的横坐标，它可以是标量、向量或矩阵，而得出的 $\boldsymbol{y}_1$ 是在这一组插值点处的插值结果。插值方法一般可以选默认的 'linear' (线性插值，它在两个样本点间简单地采用直线拟合)，'nearest' (最近点等值方式)、'cubic' (三次 Hermite 插值，当前版本的 MATLAB 中改为 'pchip') 和 'spline' (三次分段样条插值) 等，一般建议使用三次样条插值。

【例 8-1】假设已知的数据点来自函数 $f(x) = (x^2 - 3x + 5)\mathrm{e}^{-5x}\sin x$，试根据生成的数据进行插值处理，得出较平滑的曲线。

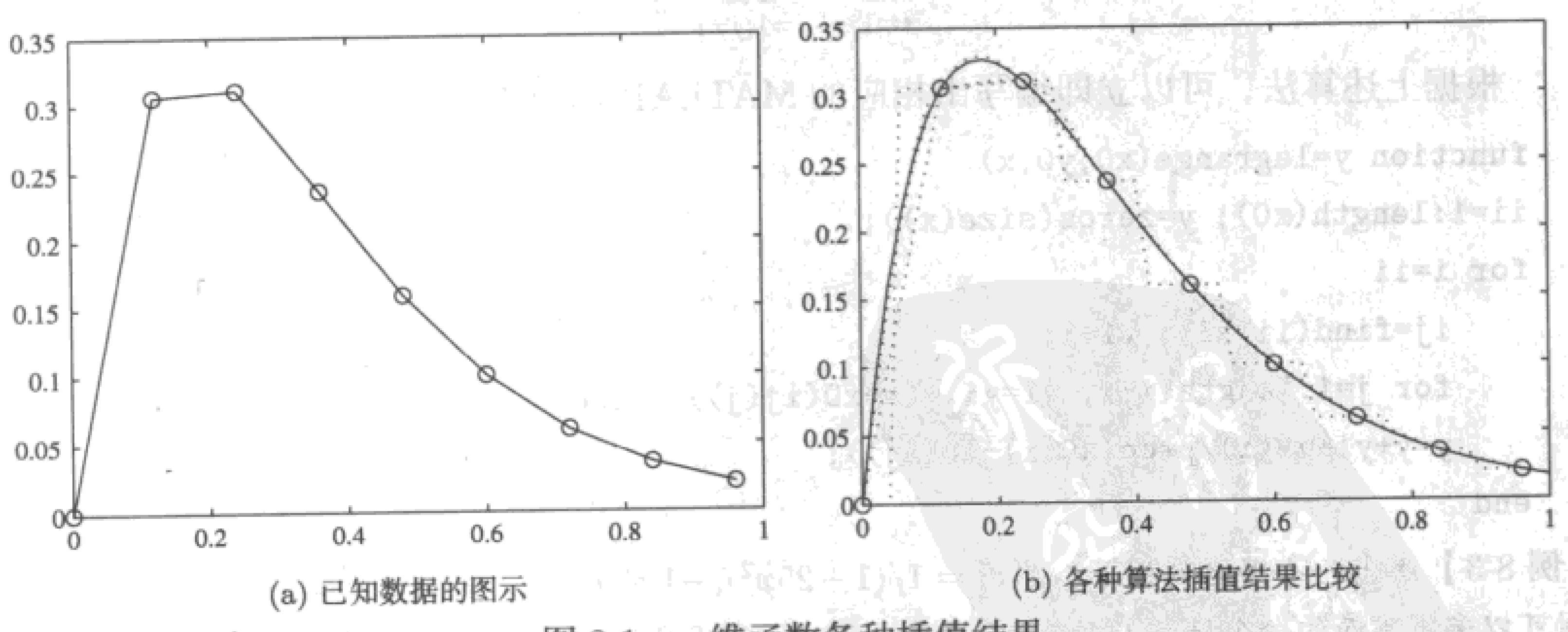
【求解】根据给出函数则可以直接生成数据，并绘制出如图 8-1 (a) 所示的折线图。

```
>> x=0:.12:1;
   y=(x.^2-3*x+5).*exp(-5*x).*sin(x); plot(x,y,x,y,'o')
```

可以看出，由这样的数据直接连线绘制出来的曲线十分粗糙，可以再选择一组插值点，然后直接调用 interp1() 函数进行插值近似。

```
>> x1=0:.02:1; y0=(x1.^2-3*x1+5).*exp(-5*x1).*sin(x1);
   y1=interp1(x,y,x1); y2=interp1(x,y,x1,'cubic');
   y3=interp1(x,y,x1,'spline'); y4=interp1(x,y,x1,'nearest');
   plot(x1,[y1',y2',y3',y4'],':',x,y,'o',x1,y0)
   [max(abs(y0(1:49)-y2(1:49))),max(abs(y0-y3)),max(abs(y0-y4))]
 ans =
   0.01765138386097    0.00861395506624    0.15982601354162
```

分别选择各种拟合选项，可以得出拟合结果与理论曲线，它们之间的比较如图 8-1 (b) 所示。



(a) 已知数据的图示　　(b) 各种算法插值结果比较

图 8-1　一维函数各种插值结果

可以看出，默认的直线型拟合得到的曲线和图 8-1 (a) 中的同样粗糙，因为该方法就是对各个点的直接连线，而 'nearest' 选项得出的拟合效果就更差了。采用 'cubic' 和 'spline' 选项的拟合更接近于理论值。事实上，应用样条插值算法得出的插值十分逼近理论值，甚至用肉眼难

以分辨。所以样条函数插值在一维数据插值拟合中还是很有效的。样条插值还可以通过 spline() 函数和样条插值工具箱求出。

【例 8-2】试编写一段程序，允许用户利用插值方法手工绘制一条光滑的曲线。

【求解】在实际应用中经常需要用户自己选定几个点，然后就能绘制出一条光滑的曲线。选择点的方法可以由 MATLAB 下的 ginput() 函数实现，有了这些点，则可以编写出下面的函数，该函数即能实现所需的功能。

```
function sketcher(vis)
x=[]; y=[]; i=1; h=[]; axis([0,1 0 1])
while 1
   [x0,y0,but]=ginput(1);
   if but==1,
      x=[x,x0]; y=[y,y0];
      h(i)=line(x0,y0); set(h(i),'Marker','o'); i=i+1;
   else, break; end
end
if nargin==1, delete(h); end  % 若需要，可以删除样本点标识
xx=[x(1):(x(end)-x(1))/100: x(end)];
yy=interp1(x,y,xx,'spline'); line(xx,yy)
```

8.1.1.2 Lagrange 插值算法及应用

Lagrange 插值算法是一般代数插值教材中经常介绍的一类插值算法[22]，对已知的 x_i, y_i 点，可以求出 $\boldsymbol{x}$ 向量上各点处的插值为

$$\phi(\boldsymbol{x})=\sum_{i=1}^{N} y_i \prod_{j=1,j\neq i}^{N} \frac{\boldsymbol{x}-x_j}{(x_i-x_j)} \tag{8-1-1}$$

根据上述算法，可以立即编写出相应的 MATLAB 函数为

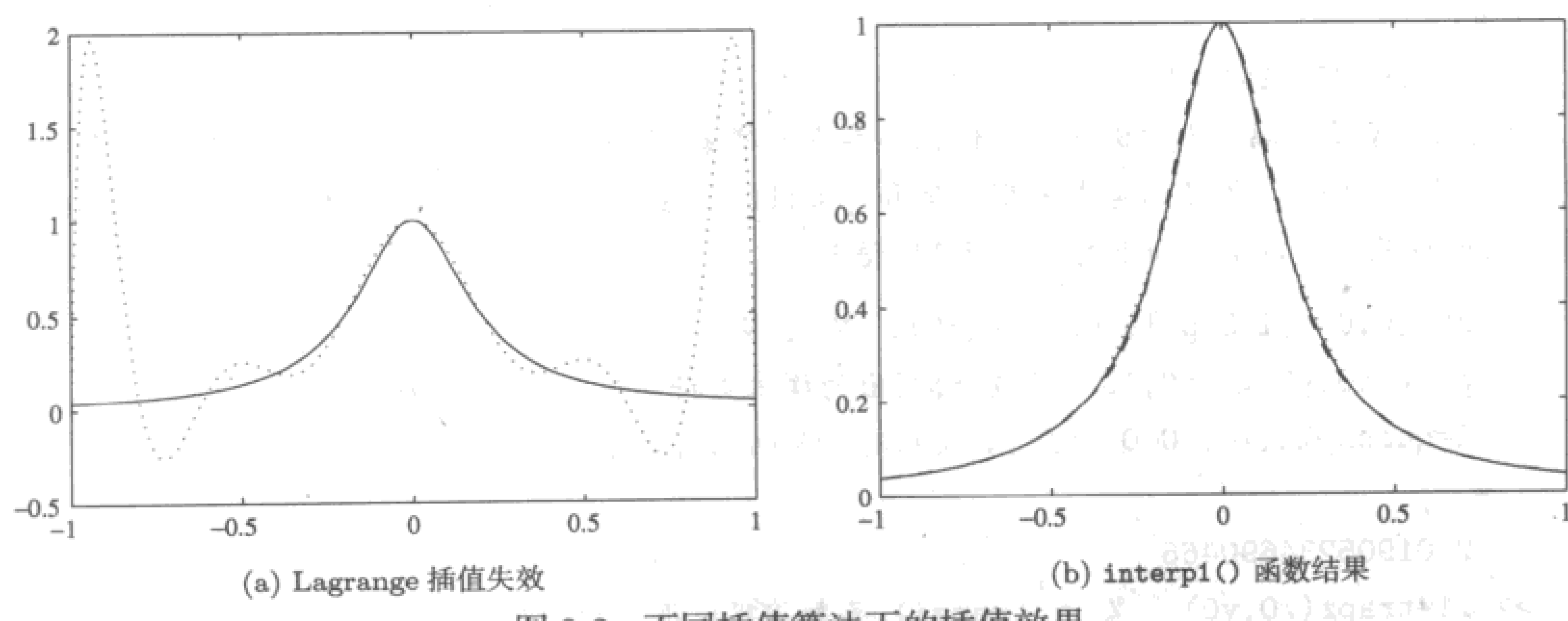
```
function y=lagrange(x0,y0,x)
ii=1:length(x0); y=zeros(size(x));
for i=ii
    ij=find(ii~=i); y1=1;
    for j=1:length(ij), y1=y1.*(x-x0(ij(j))); end
    y=y+y1*y0(i)/prod(x0(i)-x0(ij));
end
```

【例 8-3】考虑一个著名的例子，$f(x)=1/(1+25x^2), -1\leqslant x\leqslant 1$，假设已知其中一些点的坐标，则可以采用下面的命令进行 Lagrange 插值，得出如图 8-2 (a) 所示的插值曲线。

```
>> x0=-1+2*[0:10]/10; y0=1./(1+25*x0.^2);
   x=-1:.01:1; y=lagrange(x0,y0,x);  % Lagrange 插值
   ya=1./(1+25*x.^2); plot(x,ya,x,y,':')
```

由得出的插值曲线可见，用 Lagrange 插值得出的效果和精确值相差甚远，这种多项式阶次越高越发散现象的又称为 Runge 现象。所以对这个例子来说，传统的 Lagrange 算法失效。现在考虑 MATLAB 下的 interp1() 函数来解决同样的问题，通过下面的语句可以得出三次及样条插值的结果，并将各种插值结果与精确值绘制在相同的坐标系下，如图 8-2 (b) 所示。可见，用 MATLAB 中提供的算法不存在 Runge 现象，一般可以放心大胆地直接使用。

```
>> y1=interp1(x0,y0,x,'cubic'); y2=interp1(x0,y0,x,'spline');
   plot(x,ya,x,y1,':',x,y2,'--')
```



(a) Lagrange 插值失效　　(b) interp1() 函数结果

图 8-2　不同插值算法下的插值效果

8.1.2　已知样本点的定积分计算

第 3.4.1 节中给出了由样本点求解定积分的梯形方法及现成的 MATLAB 函数 trapz()。然而由例 3-28 中给出的结果看，若已知的样本点较稀疏，则得出的定积分近似将有很大的误差。如果被积函数用样条插值的方法从给定样本点直接求取，则可以编写出如下的 MATLAB 函数：

```
function y=quadspln(x0,y0,a,b)
f=inline('interp1(x0,y0,x,''spline'')','x','x0','y0');
y=quadl(f,a,b,1e-8,[],x0,y0);
```

该函数的调用格式为

I=quadspln(x_0,y_0,a,b)

其中，x_0, y_0 为样本点构成的横纵坐标向量，a, b 为积分区间，调用该函数将得出所需的定积分值，下面将通过例子演示该函数的应用。

【例 8-4】 试利用样条插值算法求解例 3-28 中给出的积分问题。

【求解】 由原题知正弦函数积分的理论值为 2，直接采用梯形法由数据求积分实际上是近似成折线与 x-轴围成区域的面积求取问题，若步长较大，则近似精度较差。这里将考虑在只已知样本点的前提下利用插值方式描述被积函数，进行积分求解的方法。由下面的语句即可得出定积分的值。

```
>> x0=0:pi/30:pi; y0=sin(x0); I=quadspln(x0,y0,0,pi)
```

```
I =
    1.99999970244699
```

可见，这样的积分结果远比例 3-28 中用梯形法得出的结果精度高得多。如果给定的样本点更稀疏，则下面可以比较梯形法和插值法的优劣就更明显了。

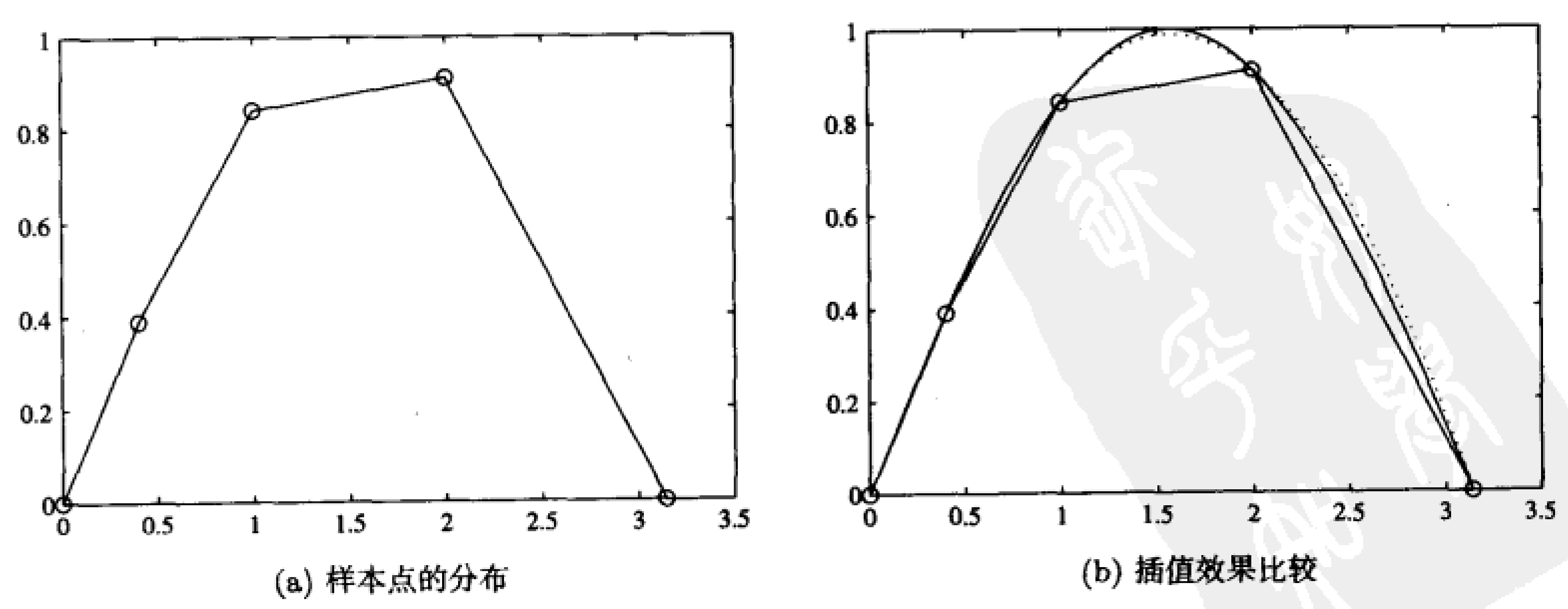
```
>> x0=0:pi/10:pi; y0=sin(x0); I1=trapz(x0,y0)
I1 =
    1.98352353750945
>> I=quadspln(x0,y0,0,pi)
I =
    2.00000015925845
```

现在考虑更极端一点的例子，即使已知再少的样本点，例如在 $x \in [0,\pi]$ 区间内仅已知 5 个不均匀分布的样本点，仍可以考虑采用插值和 quadl() 函数结合的方法求取积分值，而这时梯形法有很大的误差。可以给出如下的 MATLAB 语句：

```
>> x0=[0,0.4,1 2,pi]; y0=sin(x0);  % 生成样本点
   plot(x0,y0,x0,y0,'o')    % 绘制出的样本点折线如图 8-3 (a) 所示。
   I=quadspln(x0,y0,0,pi)  % 大约有 1% 的相对误差，应该说是相当精确的
I =
    2.01905234690466
>> I1=trapz(x0,y0)    % 用 trapz() 函数将得出很大的相对误差 (7.9%)
I1 =
    1.84155830360963
```

事实上，即使在这样稀疏的样本点下，也可以用样条插值法得出相当好的拟合效果。用下面的 MATLAB 语句可以绘制出样条插值的结果于理论之间的比较，如图 8-3 (b) 所示，其中曲线的实线部分表示原函数，虚线表示插值效果。

```
>> x=[0:0.01:pi,pi]; y0a=sin(x); y=interp1(x0,y0,x,'spline');
   plot(x0,y0,x,y,':',x,y0a,x0,y0,'o')
```



(a) 样本点的分布　　(b) 插值效果比较

图 8-3　样本点极稀疏时的插值效果

【例 8-5】仍然考虑例 3-29 中的振荡函数，假设已知其中的 150 个数据点，试采用 quadspln() 函数计算出该定积分的值，并检验其精度。

【求解】因为这里假设原函数未知，只已知数据点，所以用解析积分的算法是不可行的。若想求出该积分的数值解，则可以给出下面的指令：

```
>> x=[0:3*pi/2/200:3*pi/2]; y=cos(15*x); I=quadspln(x,y,0,3*pi/2)
I =
    0.06667117837770
```

可见，这样的结果还是很精确的。下面可以绘制出原始函数和插值曲线，如图 8-4 所示。可以看出，这样的曲线拟合效果还是很好的，从图形上和理论曲线基本看不出区别。

```
>> x0=[0:3*pi/2/1000:3*pi/2]; y0=cos(15*x0);
   y1=interp1(x,y,x0,'spline'); plot(x,y,x0,y1,':')
```

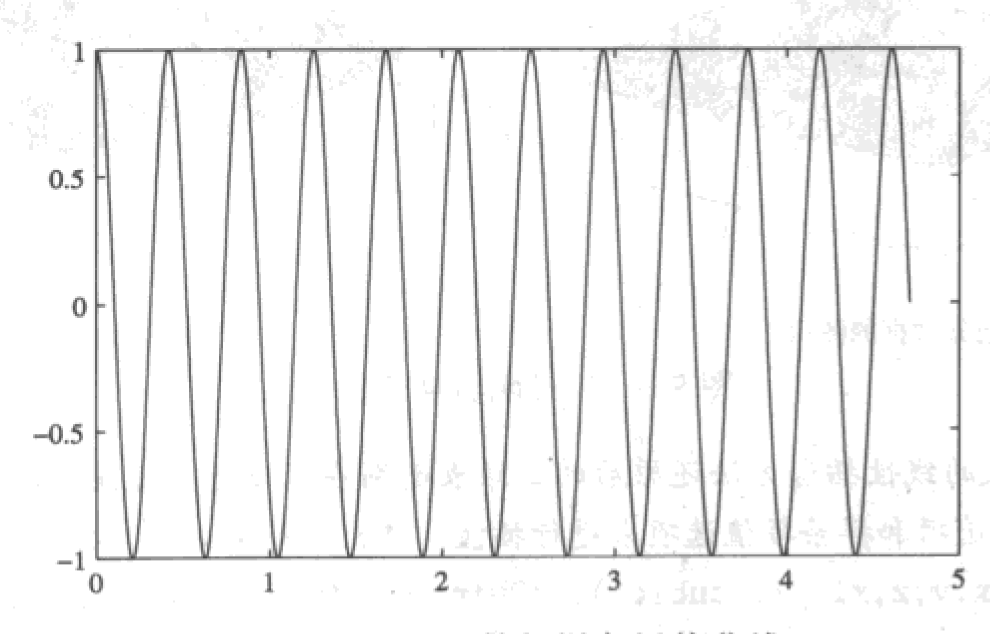


图 8-4 原型函数与样条插值曲线

对此例子来说，由于被积函数本身变化较大，给定的样本点相对较少，所以未能提供充足的信息量，来获得更高精度的积分值。若想进一步提高积分的精度，则惟一解决的途径是提高更密的样本点。

8.1.3 二维网格数据的插值问题

MATLAB 下提供了二维插值的函数，如 interp2()，该函数的调用格式为

```
z1=interp2(x0,y0,z0,x1,y1,'方法')
```

其中 x_0, y_0, z_0 为已知的数据，而 x_1, y_1 为插值点构成的新的网格参数，返回的 z_1 矩阵为在所选插值网格点处的函数近似值。interp2() 函数中可以选择的插值方法仍然为 'linear'、'cubic' 和 'spline'。和一元函数插值类似，其中最好的方法还是样条插值 'spline'，本节仍将通过例子演示、比较各种算法。

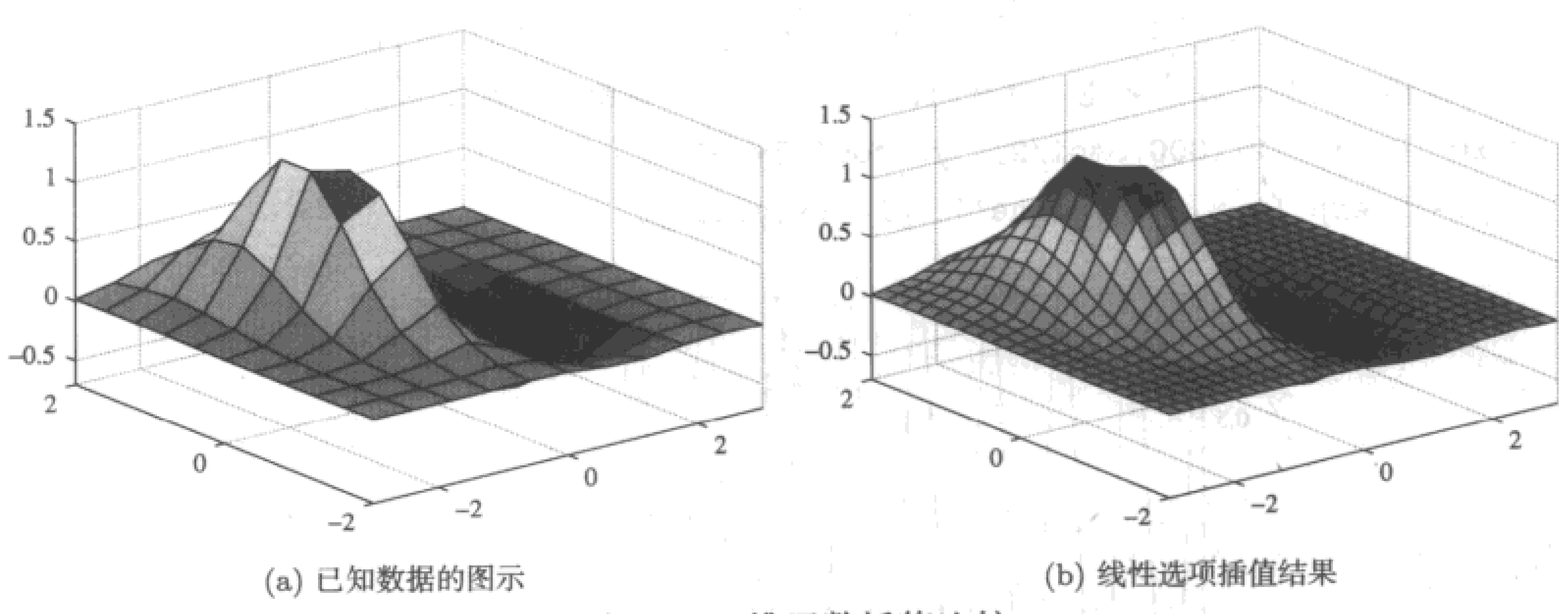
【例 8-6】假设由二元函数 $z = f(x, y) = (x^2 - 2x)\mathrm{e}^{-x^2-y^2-xy}$ 可以计算出一些较稀疏的网格数据，试根据这些数据对整个函数曲面进行各种插值拟合，并比较拟合结果。

【求解】考虑给出的二元函数，假设仅知其中较少的数据，则可以由下面命令绘制出已知数据的网格图，如图 8-5 (a) 所示。从图 8-5 (a) 可以看出，由这些数据绘制的图形还是很粗糙的。

```
>> [x,y]=meshgrid(-3:.6:3, -2:.4:2); z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
   surf(x,y,z), axis([-3,3,-2,2,-0.7,1.5])
```

选较密的插值点，则可以用下面的 MATLAB 语句采用默认的插值算法进行插值，得出的结果如图 8-5 (b) 所示。

```
>> [x1,y1]=meshgrid(-3:.2:3, -2:.2:2);
   z1=interp2(x,y,z,x1,y1); surf(x1,y1,z1), axis([-3,3,-2,2,-0.7,1.5])
```



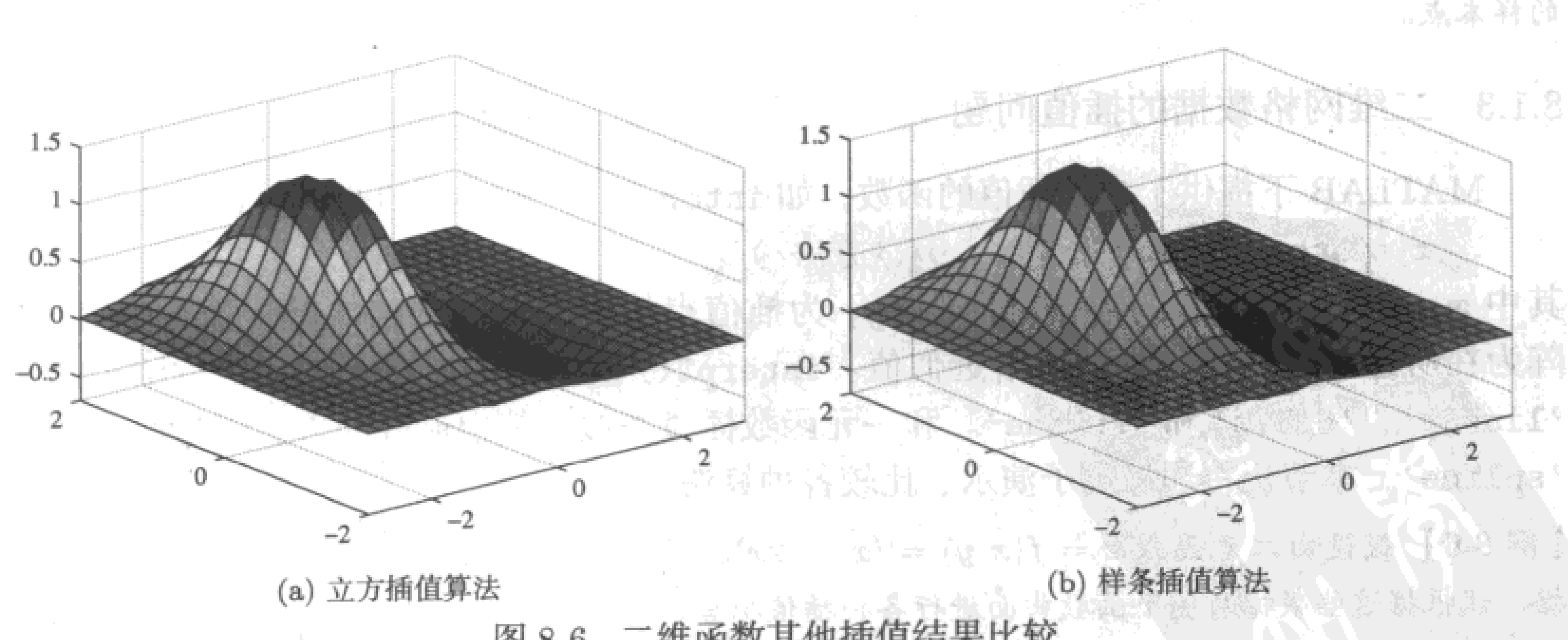
(a) 已知数据的图示　　(b) 线性选项插值结果

图 8-5　二维函数插值比较

可以看出，默认的线性插值方法还原后的三维表面图在很多地方还是太粗糙。可以用下面的命令分别由立方插值选项和样条插值选项来进行插值，得出的结果如图 8-6 所示。

```
>> z1=interp2(x,y,z,x1,y1,'cubic'); z2=interp2(x,y,z,x1,y1,'spline');
   surf(x1,y1,z1), axis([-3,3,-2,2,-0.7,1.5])
   figure; surf(x1,y1,z2), axis([-3,3,-2,2,-0.7,1.5])
```

可以看出，这样的插值效果还都是比较理想的。



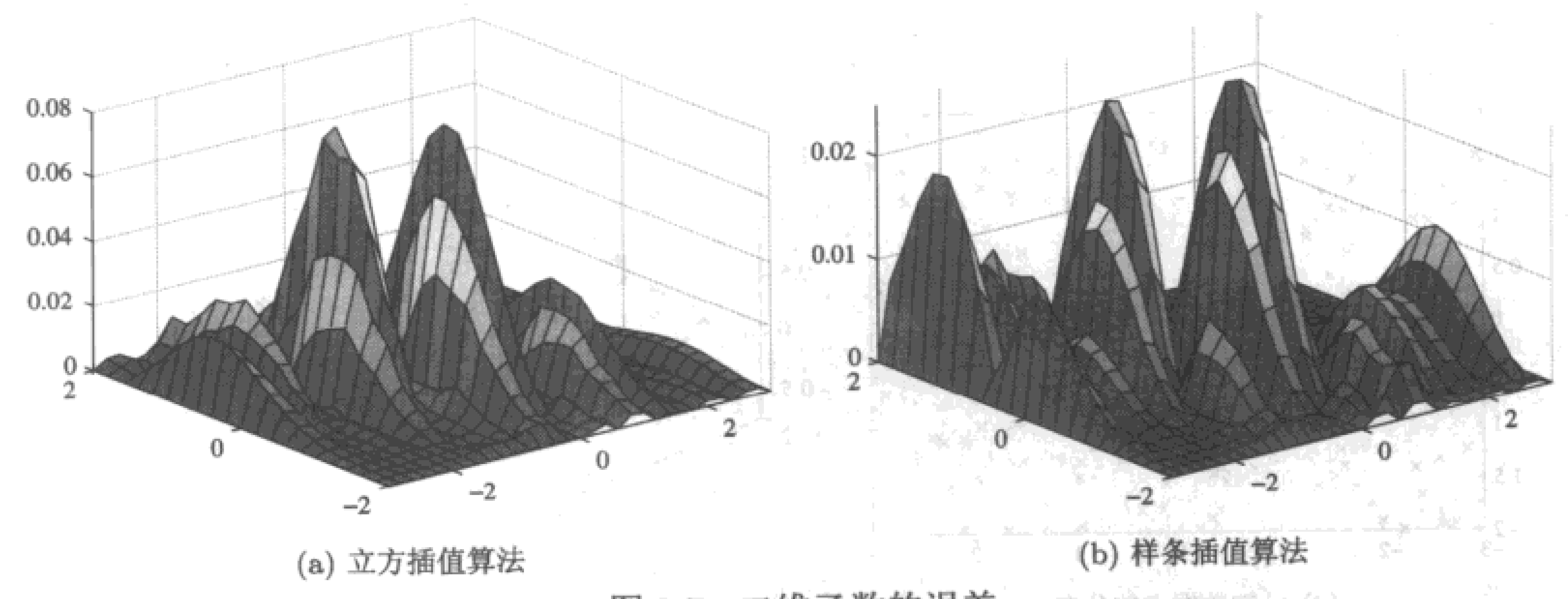
(a) 立方插值算法　　(b) 样条插值算法

图 8-6　二维函数其他插值结果比较

通过下面的误差分析还可以进一步比较两种算法，因为网格已知，故可以由已知函数计算出

z 的精确值，所以可以通过下面的语句求出两种算法得出的矩阵 z_1 和 z_2 与真值 z 之间误差的绝对值，分别如图 8-7 (a)、图 8-7 (b) 所示。可以看出，选择样条方法的插值精度要远高于立方插值算法，所以在实际应用中建议使用 'spline' 插值选项。

```
>> z=(x1.^2-2*x1).*exp(-x1.^2-y1.^2-x1.*y1);  % 新网格各点的函数值
   surf(x1,y1,abs(z-z1)), axis([-3,3,-2,2,0,0.08])
   figure; surf(x1,y1,abs(z-z2)), axis([-3,3,-2,2,0,0.025])
```



(a) 立方插值算法　　(b) 样条插值算法

图 8-7　二维函数的误差

8.1.4　二维一般分布数据的插值问题

通过上面的例子可以看出，MATLAB 提供的二维插值函数还是能较好地进行二维插值运算的。但该函数有一个重要的缺陷，就是它只能处理以网格形式给出的数据，如果已知数据不是以网格形式给出的，则用该函数是无能为力的。在实际应用中，大部分问题都是以实测的多组 (x_i, y_i, z_i) 点给出的，所以不能直接使用函数 interp2() 进行二维插值。

MATLAB 语言中提供了一个更一般的 griddata() 函数，用来专门解决这样的问题。该函数的调用格式为

z=griddata(x_0,y_0,z_0, x,y,'v4')

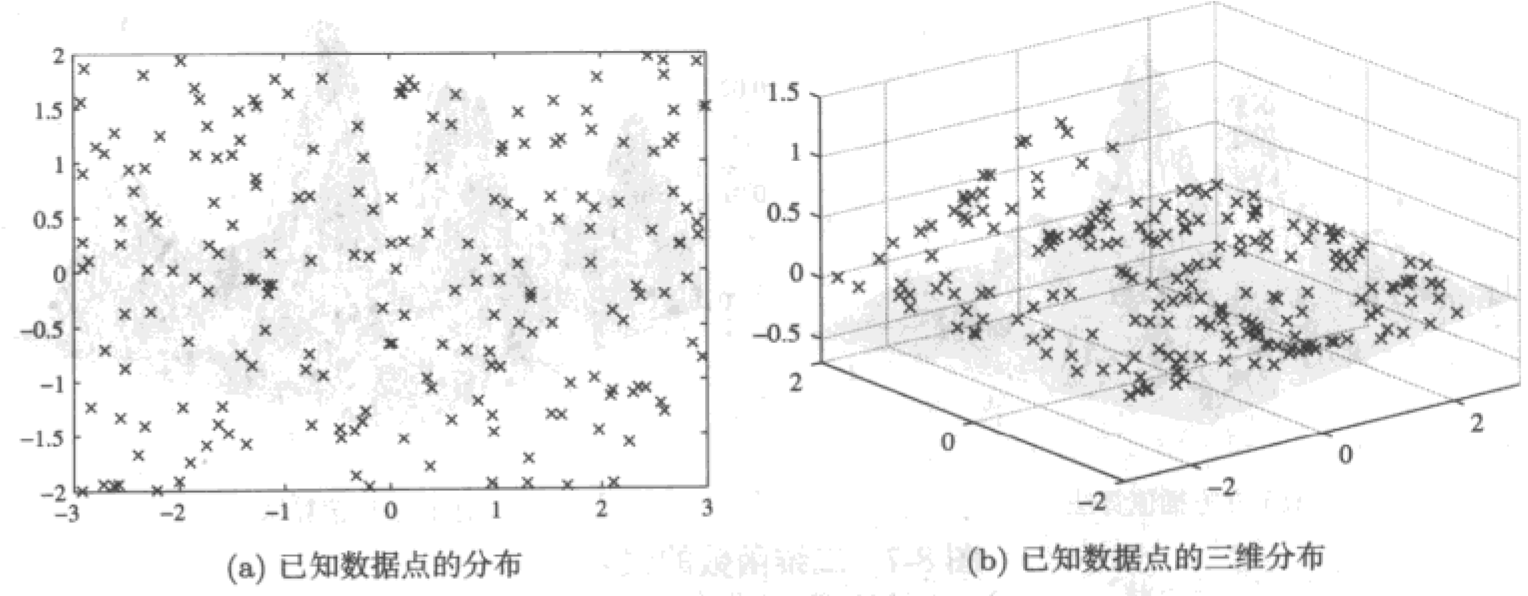
其中，x_0, y_0, z_0 是已知的样本点坐标，这里并不要求是网格型的，可以是任意分布的，均由向量给出。x, y 是期望的插值位置，可以是单个点，可以是向量或网格型矩阵，得出的 z 应该维数和 x, y 一致，表示插值的结果。选项 'v4' 是 MATLAB 4.0 版本中提供的插值算法，公认效果较好，但没有一个正式的名称，所以这里用 'v4' 表明该算法。除了 'v4' 选项外，还可以使用 'linear'、'cubic' 和 'nearest' 等算法。

【例 8-7】仍考虑原型函数 $z = f(x,y) = (x^2 - 2x)\mathrm{e}^{-x^2-y^2-xy}$，在 $x \in [= 3,3]$, $y \in [-2,2]$ 矩形区域内随机选择一组 (x_i, y_i) 坐标，就可以生成一组 z_i 的值。以这些值为已知数据，用一般分布数据插值函数 griddata() 进行插值处理，并进行误差分析。

【求解】这里选择 200 个随机数构成的点，则可以用下面的语句生成 x, y, z 向量，但由于这些数

据不是网格数据，所以得出的数据向量不能直接用三维曲面的形式表示。但可以通过下面的语句将各个样本点在 x-y 平面上的分布形式显示出来，如图 8-8 (a) 所示，也可以绘制出样本点的三维分布，如图8-8 (b) 所示。可以看出，这些分布点还是比较均匀的。

```
>> x=-3+6*rand(200,1); y=-2+4*rand(200,1);
   z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);   % 生成已知数据
   plot(x,y,'x')   % 样本点的二维分布
   figure, plot3(x,y,z,'x'), axis([-3,3,-2,2,-0.7,1.5])
```

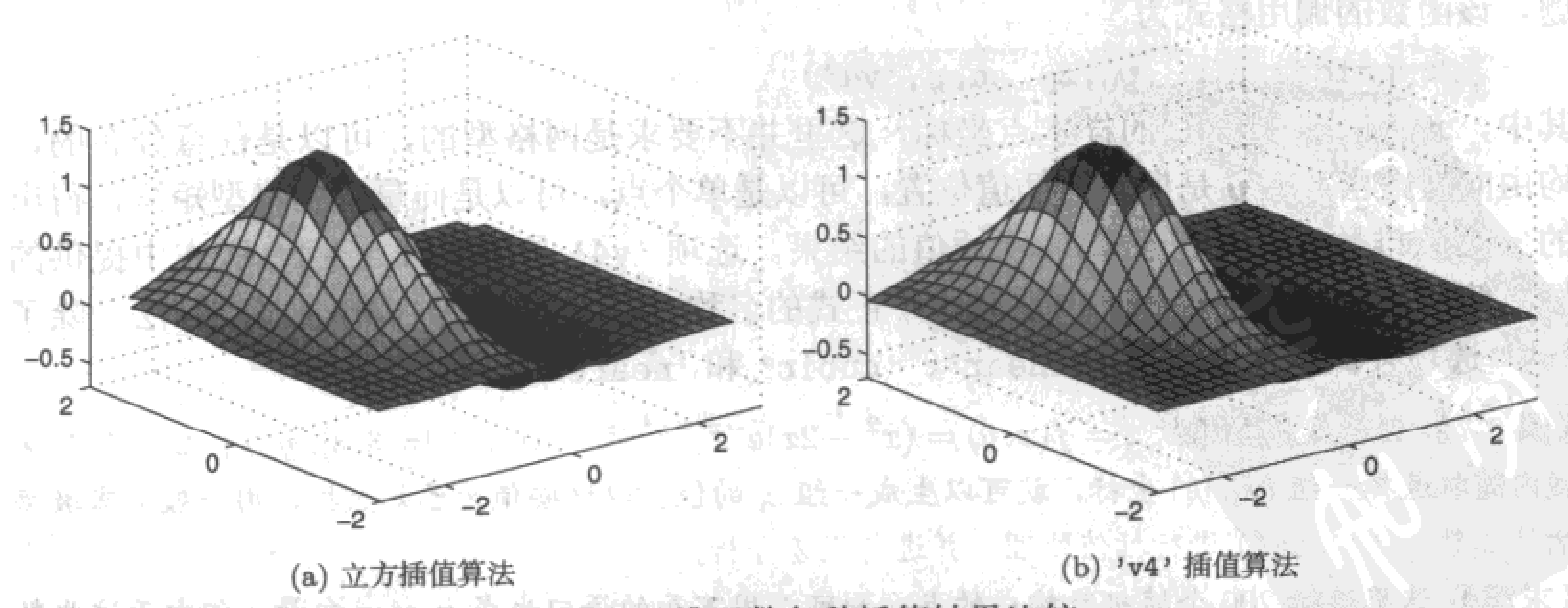


(a) 已知数据点的分布 (b) 已知数据点的三维分布

图 8-8 已知样本数据显示

仍选定按照例 8-6 中给出的方法生成网格矩阵，则可以用 'cubic' 和 'v4' 两种算法获得插值结果，还可以绘制出拟合后的曲面形式，分别如图 8-9 (a)、图 8-9 (b) 所示。可以看出，用 'v4' 算法得出的结果效果明显更好些，用 'cubic' 插值算法得出的曲面残缺不全。

```
>> [x1,y1]=meshgrid(-3:.2:3, -2:.2:2);
   z1=griddata(x,y,z,x1,y1,'cubic'); surf(x1,y1,z1),
   axis([-3,3,-2,2,-0.7,1.5]); z2=griddata(x,y,z,x1,y1,'v4');
   figure; surf(x1,y1,z2), axis([-3,3,-2,2,-0.7,1.5])
```

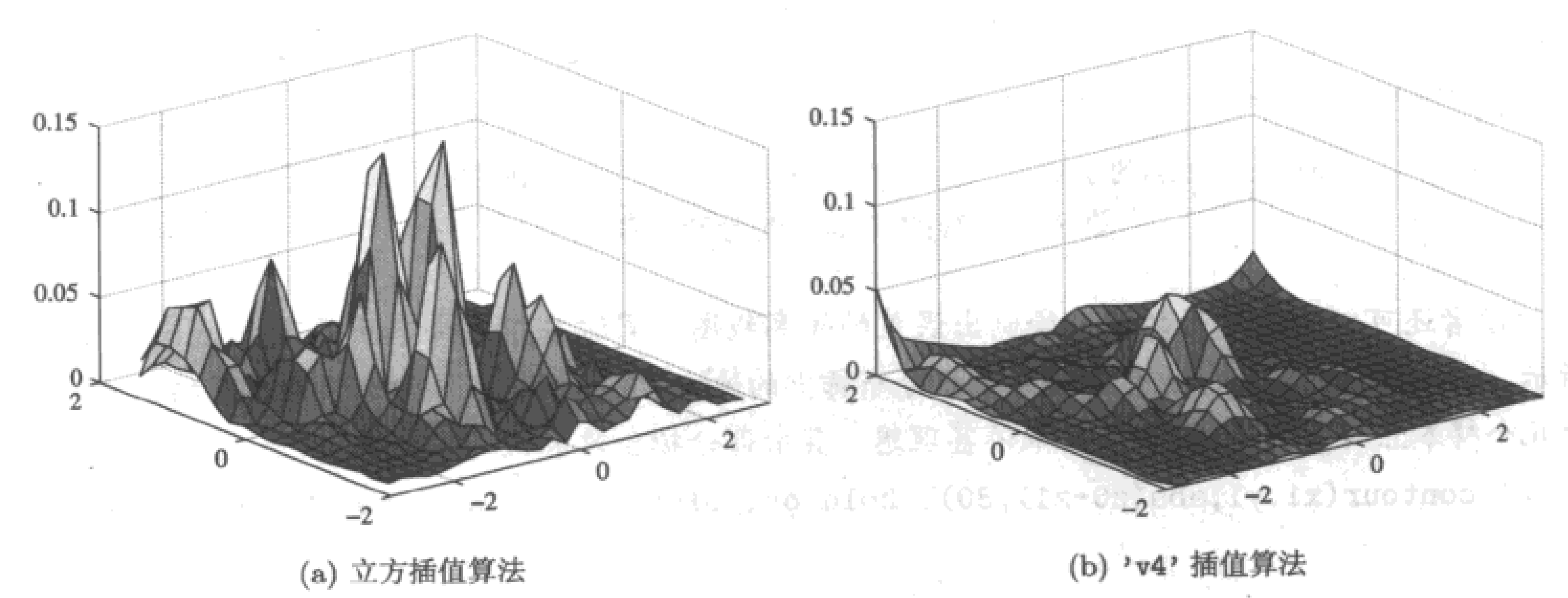


(a) 立方插值算法 (b) 'v4' 插值算法

图 8-9 二维函数各种插值结果比较

还可以进一步进行误差分析。用下面的语句可以先计算出在新网格点处函数值的精确解，并用这些点和两种方法计算出来的误差，得出如图 8-10 (a)、图 8-10 (b) 所示的误差曲面。可见，用'v4'选项的插值结果明显优于立方插值算法，所以在实际应用中建议采用该算法。

```
>> z0=(x1.^2-2*x1).*exp(-x1.^2-y1.^2-x1.*y1);  % 新网格各点的函数值
   surf(x1,y1,abs(z0-z1));  axis([-3,3,-2,2,0,0.15])
   figure; surf(x1,y1,abs(z0-z2));  axis([-3,3,-2,2,0,0.15])
```



(a) 立方插值算法　　(b) 'v4' 插值算法

图 8-10　二维函数插值误差分析

【例 8-8】 前面已经提及，给定的样本点在 x-y 平面分布较均匀，现在人为剔除某些点，表明已知数据分布不均匀，这时再进行插值分析，观察插值效果。

【求解】 由已知的 x,y,z 矩阵人为地剔除在以 $(-1,-1/2)$ 点为圆心，以 0.5 为半径的圆内的点，则可以采用下面语句：

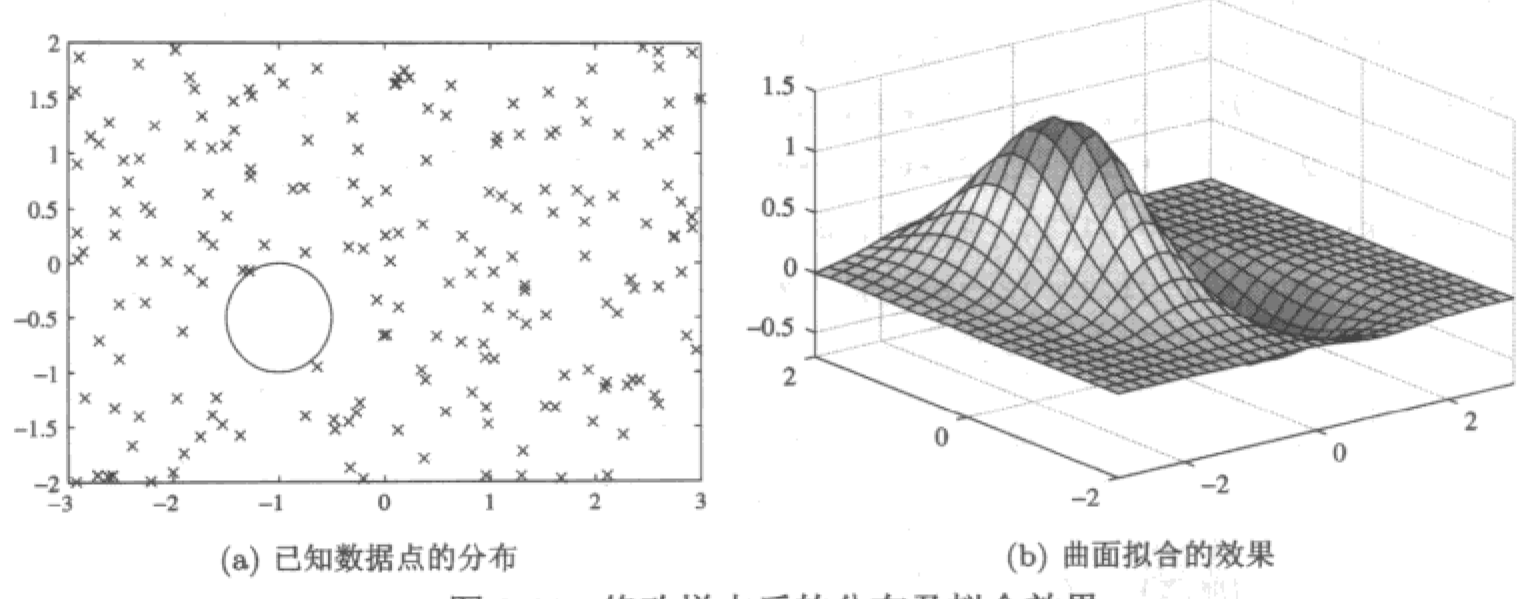
```
>> x=-3+6*rand(200,1); y=-2+4*rand(200,1);  % 重新生成样本点
   z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
   ii=find((x+1).^2+(y+0.5).^2>0.5^2);  % 找出不满足条件的点坐标
   x=x(ii); y=y(ii); z=z(ii); plot(x,y,'x')
   t=[0:.1:2*pi,2*pi]; x0=-1+0.5*cos(t); y0=-0.5+0.5*sin(t);
   line(x0,y0)    % 在图形上叠印该圆，可见，圆内样本点均已剔除
```

这时将得出如图 8-11 (a) 所示的样本点分布图，同时还叠印了圆。可见，在该圆内确实样本点均已经剔除。用新的样本点可以拟合出曲面，如图 8-11 (b) 所示。可见，拟合效果还是很好的。

```
>> [x1,y1]=meshgrid(-3:.2:3, -2:.2:2);
   z1=griddata(x,y,z,x1,y1,'v4');
   surf(x1,y1,z1), axis([-3,3,-2,2,-0.7,1.5])
```

现在可以进行误差分析了，用下面的语句可以得出误差，并绘制出误差曲面，如图 8-12 (a) 所示，可以看出，尽管原始样本点数据被人为剔除掉了一个较大的区域，但拟合效果还是很好。

```
>> z0=(x1.^2-2*x1).*exp(-x1.^2-y1.^2-x1.*y1);
   surf(x1,y1,abs(z0-z1)), axis([-3,3,-2,2,0,0.1])
```

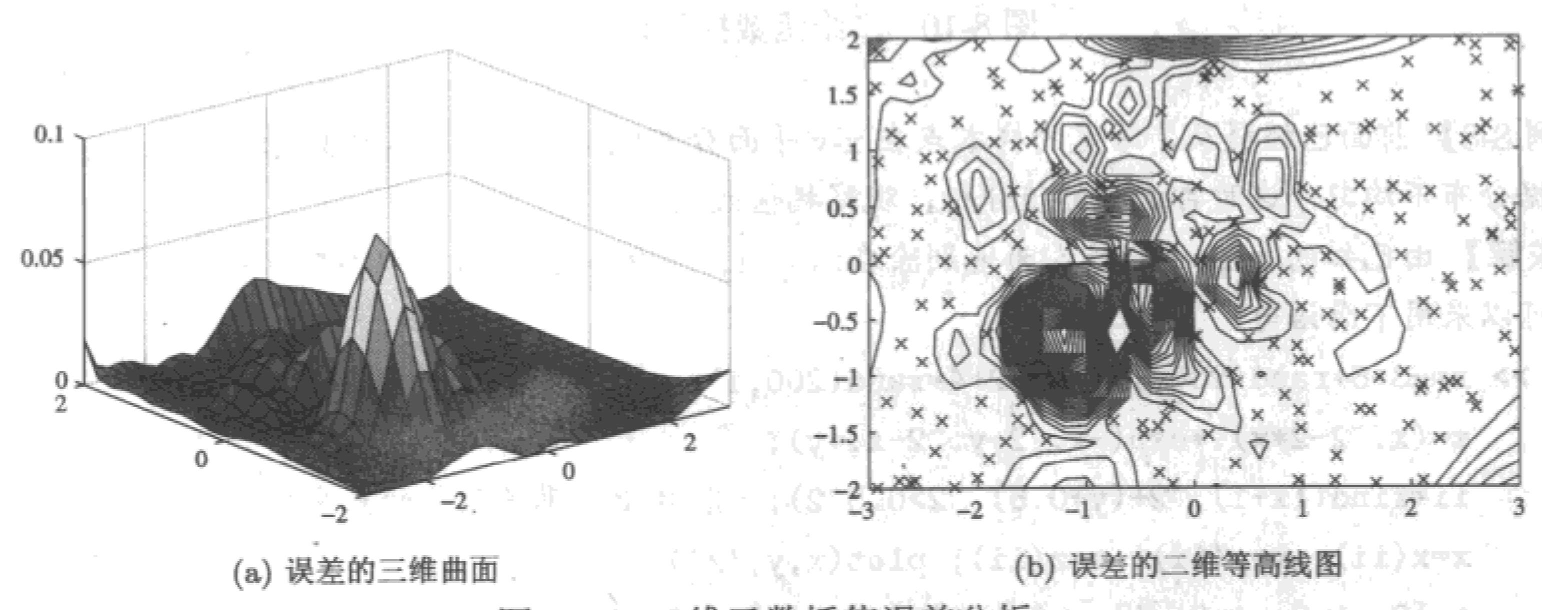


(a) 已知数据点的分布　　(b) 曲面拟合的效果

图 8-11　修改样本后的分布及拟合效果

读者还可以给出下面的语句绘制出误差的等高线图，同时叠印出样本点分布，如图 8-12 (b) 所示。从图中可用看出，原始样本点数据分布稀少的地方 (在本例子中人为剔除样本点的区域和其他几个样本点稀少的区域) 拟合效果不甚理想，其余部分拟合效果较好。

```
>> contour(x1,y1,abs(z0-z1),30); hold on, plot(x,y,'x'); line(x0,y0)
```



(a) 误差的三维曲面　　(b) 误差的二维等高线图

图 8-12　二维函数插值误差分析

由此可以得出结论，数据插值拟合效果的好坏很大程度上取决于数据点的分布情况，如果某个区域的数据点分布较少，则很难通过插值的方式恢复该区域，因为这个区域已知的信息量是不足以高精度恢复数据的，所以为使得该函数的拟合精度较高，还是建议在数据采集时均匀地多选择一些点。

8.1.5　高维插值问题

三维的网格数据生成仍然可以用 meshgrid() 函数实现。该函数的具体调用格式为

$[\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}]$=meshgrid($\boldsymbol{x}_1,\boldsymbol{y}_1,\boldsymbol{z}_1$)

其中，$\boldsymbol{x}_1,\boldsymbol{y}_1,\boldsymbol{z}_1$ 为这三维所需要的分割形式，应该是向量形式给出的，返回的 $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}$ 为

网格的数据生成，均为三维数组。

n 维网格数据的生成可以使用 ndgrid() 函数。该函数的调用格式为

$[\boldsymbol{x}_1,\cdots,\boldsymbol{x}_n]$=ndgrid($\boldsymbol{v}_1,\cdots,\boldsymbol{v}_n$)

其中，$\boldsymbol{v}_1,\cdots,\boldsymbol{v}_n$ 为这 n 维所需要的分割形式，应该是向量形式给出的，返回的 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_n$ 为网格数据生成的效果，这时返回的 $\boldsymbol{x}_i$ 为 n 维数组。

若已知按空间网格取的样本点，则可以用 interp3() 函数或更一般的 interpn() 函数进行插值运算。这些函数的调用格式和 interp2() 一致，这里不详细介绍了。若已知样本点不是以空间网格数据的形式给出，则类似地可以调用 griddata3() 或更一般的 griddatan() 函数对其进行插值拟合。这些函数的调用格式可以参见 griddata() 函数。

【例 8-9】假设已知某三元函数 $V(x,y,z)=\mathrm{e}^{x^2z+y^2x+z^2y}\cos(x^2yz+z^2yx)$，可以通过该函数生成一些网格型样本点，试根据样本点进行拟合，并给出拟合误差。

【求解】在 MATLAB 下无法绘制真正意义下的四维图形，所以这里的分析只局限于数字层面上。先调用 meshgrid() 函数生成一组三维网格坐标点，则可以求出样本点处的函数值 $\boldsymbol{V}$，从下面的结果可以看出，插值的精度较高。

```
>> [x,y,z]=meshgrid(-1:0.2:1);
   V=exp(x.^2.*z+y.^2.*x+z.^2.*y).*cos(x.^2.*y.*z+z.^2.*y.*x);
   [x0,y0,z0]=meshgrid(-1:0.05:1);
   V0=exp(x0.^2.*z0+y0.^2.*x0+z0.^2.*y0).*cos(x0.^2.*y0.*z0+z0.^2.*y0.*x0);
   V1=interp3(x,y,z,V,x0,y0,z0,'spline'); err=V1-V0; max(err(:))
ans =
    0.04186238115447
```

8.2 样条插值与数值微积分

前面介绍的插值函数是简单的插值算法。MATLAB 提供了一个样条插值工具箱，可以更好地求解样条插值问题。还可以借助于样条数据结构容易地求解微积分问题。所以本节可以认为是第 8.1 节以及第 3.3 节和第 3.4 节的拓广。

样条函数是函数逼近的一种方法，其中三次样条函数和 B 样条函数是两类常用的样条函数。下面首先分别介绍这两类样条函数的表示方法，然后介绍利用 MATLAB 的样条插值工具箱求解数值微分和数值积分的问题。

8.2.1 样条插值的 MATLAB 表示

8.2.1.1 三次样条函数及其 MATLAB 表示

三次样条函数的定义是，已知平面上 n 个点 (x_i,y_i) $(i=1,2,\cdots,n)$，其中，$x_1<x_2<\cdots<x_n$，这些点称为样本点。如果有某函数 $S(x)$ 满足下面 3 个条件，则称 $S(x)$ 为经过这 n 个点的三次样条函数。

① $S(x_i)=y_i$ $(i=1,2,\cdots,n)$，亦即该函数经过这些样本点。

② $S(x)$ 在每个子区间 $[x_i,x_{i+1}]$ 上为三次多项式

$$S(x)=c_{i1}(x-x_i)^3+c_{i2}(x-x_i)^2+c_{i3}(x-x_i)+c_{i4}$$

③ $S(x)$ 在整个区间 $[x_1,x_n]$ 上有连续的一阶及二阶导数。

MATLAB 的样条插值工具箱中提供了 csapi() 函数来定义一个三次样条函数类，其调用格式很简单，为

S=csapi($\boldsymbol{x}$,$\boldsymbol{y}$)

其中，$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]$，$\boldsymbol{y}=[y_1,y_2,\cdots,y_n]$ 为样本点，得出的 S 是一个三次样条函数对象，其成员变量包括子区间点、各个三次多项式系数等。

样条函数对象的插值结果可以由 fnplt() 绘制出来，对给定的向量 $\boldsymbol{x}_p$，也可以由 fnval() 函数计算出来。这两个函数的调用格式为

fnplt(S)， $\boldsymbol{y}_p$=fnval(S,$\boldsymbol{x}_p$)

其中得出的 $\boldsymbol{y}_p$ 为 $\boldsymbol{x}_p$ 上各点的插值结果。

【例 8-10】 试求出例 8-5 中给出稀疏数据的三次样条插值结果。

【求解】 由三次样条函数的调用语句可以立即得出给定数据的样条插值结果，并和原理论数据同时绘制出来，如图 8-13 所示。

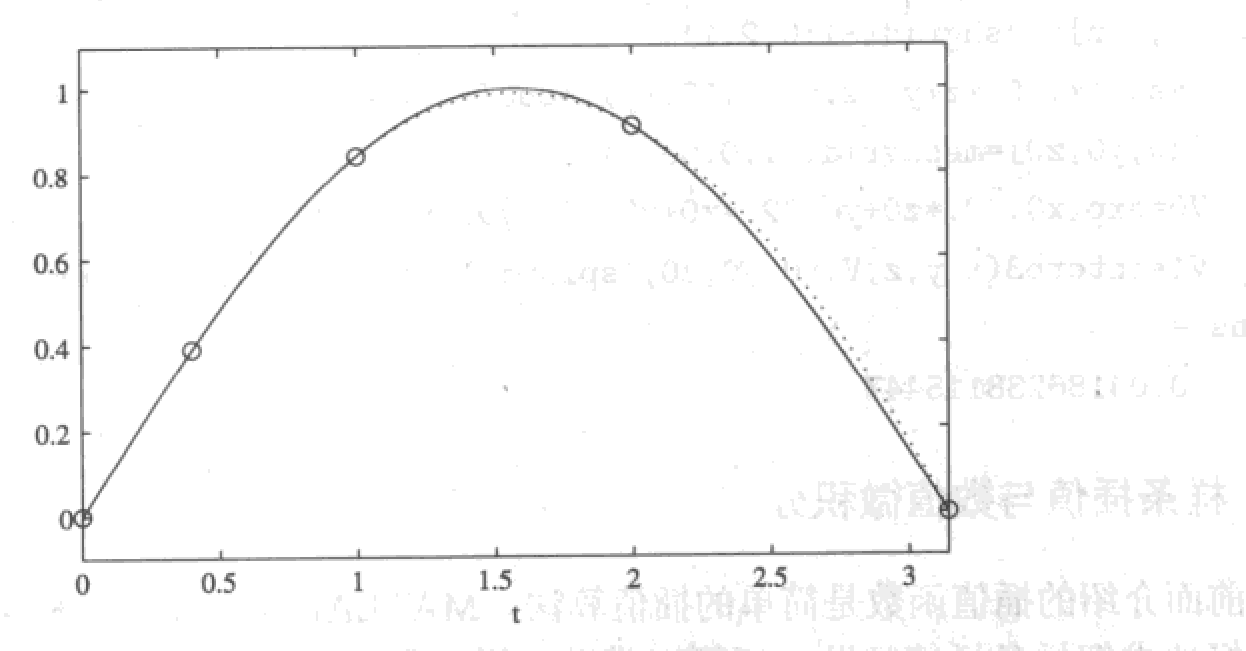


图 8-13 给定稀疏数据的三次样条插值效果

```
>> x0=[0,0.4,1 2,pi]; y0=sin(x0);
   sp=csapi(x0,y0), fnplt(sp,':'); hold on,
   ezplot('sin(t)',[0,pi]); plot(x0,y0,'o')
sp =
      form: 'pp'
    breaks: [0 0.40000000000000 1 2 3.14159265358979]
     coefs: [4x4 double]
    pieces: 4
     order: 4
       dim: 1
>> sp.coefs
ans =
```

```
-0.16265031352655   0.00758565399762   0.99653564433683                  0
-0.16265031352655  -0.18759472223424   0.92453201704218   0.38941834230865
 0.02443571684740  -0.48036528658203   0.52375601175242   0.84147098480790
 0.02443571684740  -0.40705813603983  -0.36366741086945   0.90929742682568
```

其中得出的每一行为对应区间内的三次多项式系数。例如，在 (0.4,1) 区间内，插值多项式可以表示为 $S_2(x) = -0.162650314(x-0.4)^3-0.187594722(x-0.4)^2+0.924532017(x-0.4)+0.389418342$。

【例 8-11】 试用三次样条插值的方法对例 8-1 中给出的数据进行拟合。

【求解】 用下面的语句可以建立起描述已知数据的样条插值类，并得出各段三次多项式系数，由表 8-1 给出，根据该表可以用多项式方式计算出样条插值的值。

```
>> x=0:.12:1; y=(x.^2-3*x+5).*exp(-5*x).*sin(x);
   sp=csapi(x,y); fnplt(sp)
   c=[sp.breaks(1:4)' sp.breaks(2:5)' sp.coefs(1:4,:),... % 生成表 8-1 数据
      sp.breaks(5:8)' sp.breaks(6:9)' sp.coefs(5:8,:) ];
```

表 8-1 分段三次多项式样条插值系数表

分段	三次多项式系数				分段	三次多项式系数			
区间	c_1	c_2	c_3	c_4	区间	c_1	c_2	c_3	c_4
(0, 0.12)	24.7396	-19.359	4.5151	0	(0.48, 0.6)	-0.2404	0.7652	-0.5776	0.1588
(0.12, 0.24)	24.7396	-10.4526	0.9377	0.3058	(0.6, 0.72)	-0.4774	0.6787	-0.4043	0.1001
(0.24, 0.36)	4.5071	-1.5463	-0.5022	0.3105	(0.72, 0.84)	-0.4559	0.5068	-0.2621	0.0605
(0.36, 0.48)	1.9139	0.07623	-0.6786	0.2358	(0.84, 0.96)	-0.4559	0.3427	-0.1601	0.03557

csapi() 函数还可以处理多个自变量的网格数据三次样条插值类，其调用格式为

S=csapi($\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n\}$,$\boldsymbol{z}$)

其中，$\boldsymbol{x}_i$ 为自变量的网格标志，$\boldsymbol{z}$ 网格数据的样本点，得出的 S 是三次样条函数对象。

【例 8-12】 试用三次样条插值方法得出例 8-6 中给出网格数据的样条插值拟合，并绘制出曲面。

【求解】 用下面的语句自然就能得出样条插值对象 sp，并绘制出如图 8-14 所示的曲面。可见，这样的插值结果与 interp2() 函数得出的完全一致。

```
>> x0=-3:.6:3; y0=-2:.4:2; [x,y]=ndgrid(x0,y0); % 注意这里只能用 ndgrid
   z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);   % 否则生成的 z 矩阵顺序有问题
   sp=csapi({x0,y0},z); fnplt(sp);
```

注意，这里的 $\boldsymbol{z}$ 矩阵应该是基于用 ndgrid() 函数生成的 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 矩阵，而不能用 meshgrid() 函数生成，因为用其生成的 $\boldsymbol{z}$ 矩阵数据排列方式和样条插值工具箱不一致。

8.2.1.2 B 样条函数及其 MATLAB 表示

B 样条插值为另一类常用的样条函数，其数学描述方式比较不易理解，故这里只介绍该类样条函数对象的建立函数 spapi()，而不介绍其数学描述。若已知样本点数据向量 $\boldsymbol{x}$

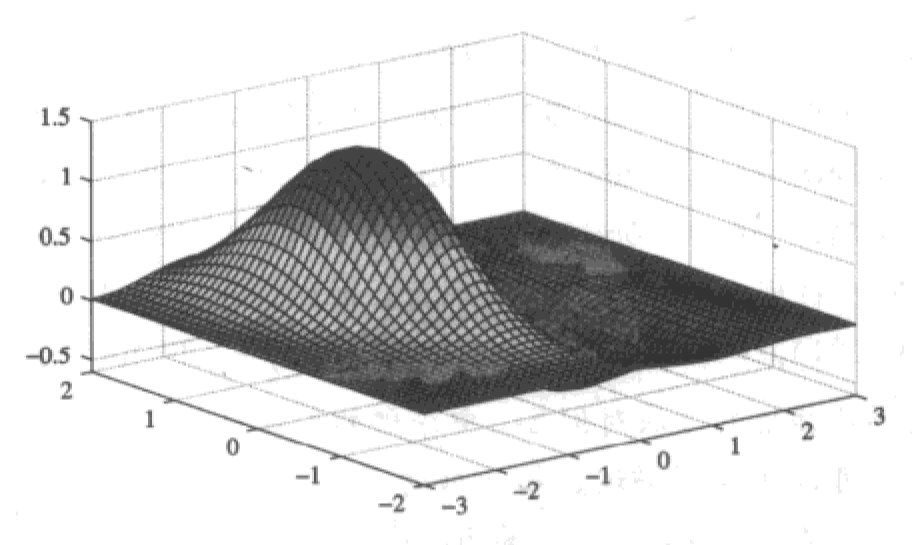


图 8-14　二维函数插值结果

和 y，则可以通过下面的语句直接建立起 B 样条插值对象 S 为

S=spapi(k,x,y)

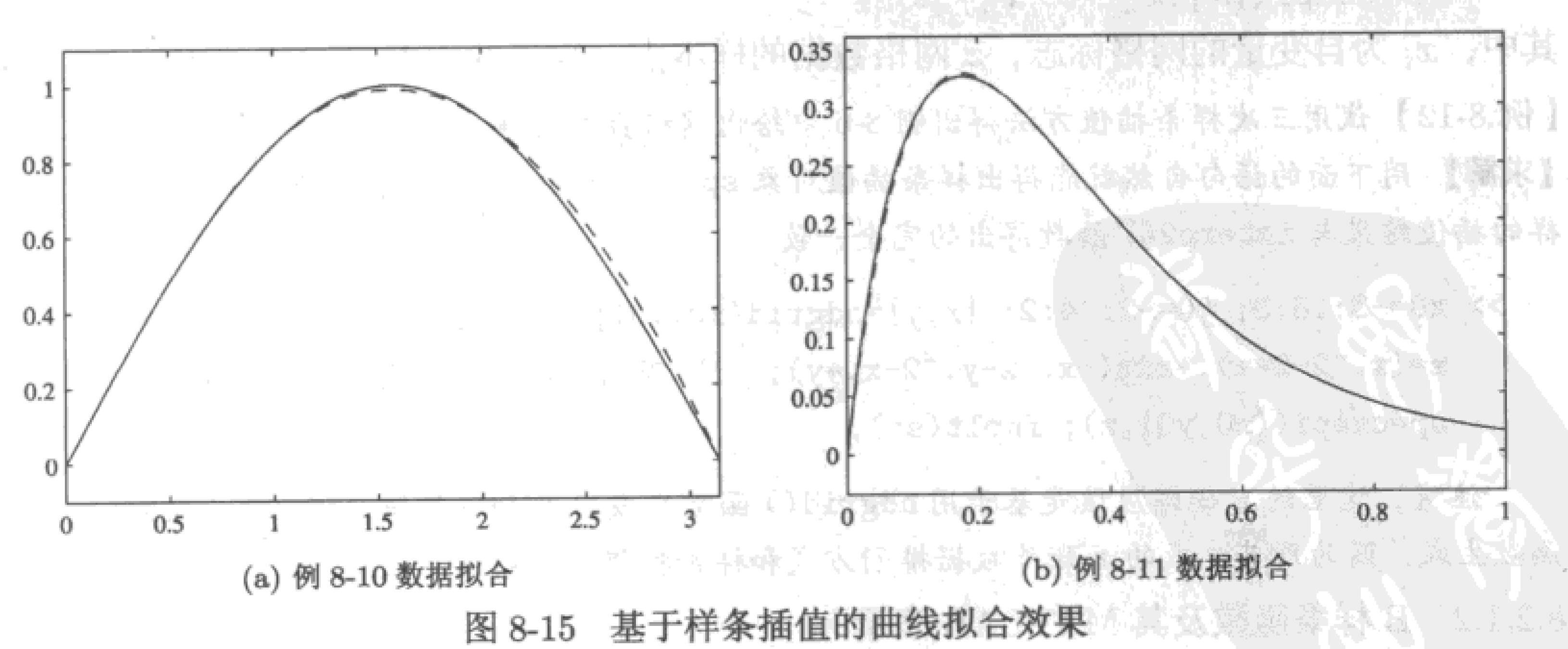
其中，k 为用户选定的 B 样条阶次。一般选择 k=4,5 能得出较好的插值效果。对某些特定的问题适当提高 k 值能改善插值效果。

【例 8-13】分别用 B 样条函数对例 8-10 和 8-11 中给出的数据进行 5 次 B 样条函数拟合，并与三次分段多项式样条函数拟合的结果相比较。

【求解】先考虑例 8-10 中给出的数据，可以用下面的语句进行拟合，得出如图 8-15 (a) 所示的拟合效果。其中的 B 样条插值效果几乎看不出和理论曲线的差异。

```
>> x0=[0,0.4,1 2,pi]; y0=sin(x0); ezplot('sin(t)',[0,pi]); hold on
   sp1=csapi(x0,y0); fnplt(sp1,'--');  % 三次分段多项式样条插值
   sp2=spapi(5,x0,y0); fnplt(sp2,':')  %5 次 B 样条插值
```

可见，用 5 次 B 样条插值的效果远远优于三次分段多项式的拟合效果。对例 8-11 中给出的数据进行拟合，将得出如图 8-15 (b) 所示的效果，B 样条亦远远优于三次样条插值。



(a) 例 8-10 数据拟合　　(b) 例 8-11 数据拟合

图 8-15　基于样条插值的曲线拟合效果

```
>> x=0:.12:1; y=(x.^2-3*x+5).*exp(-5*x).*sin(x);
   ezplot('(x^2-3*x+5)*exp(-5*x)*sin(x)',[0,1]), hold on
```

```
sp1=csapi(x,y); fnplt(sp1,'--'); sp2=spapi(5,x,y); fnplt(sp2,':')
```

8.2.2 基于样条插值的数值微积分运算

既然用样条插值的方法能由给定的数据进行曲线拟合，且在给定数据较稀疏的情形拟合效果也是很理想的，故可以利用插值对象对给定数据函数进行微积分运算。和第 3.3 与 3.4 节介绍的算法相比，基于样条插值的数值微分算法有其特色，更适用于给定样本数据较稀疏时的数值微分。从数值积分的角度看，这里得出的数值积分是数值积分的函数，亦即求取 $F(x)=\int_{x_0}^{x} f(t)\mathrm{d}t$ 的值，而不是单纯的定积分值，其中 x_0 是用户指定的积分区域左端边界值。当然，单纯的定积分也可以用积分函数得出，即 $I=F(b)-F(a)$。

8.2.2.1 基于样条插值的数值微分运算

基于样条函数的数值微分运算可以由 fnder() 函数直接计算出来，如下：

S_d=fnder(S,k),	该函数可以求取 S 的 k 阶导数
S_d=fnder(S,[$k_1,\cdots,k_n$]),	可以求取多变量函数的偏导数

该函数的两种调用方法中，前一种方法能直接求取 S 样条对象的 k 阶导数，得出的结果仍然是样条对象 S_d。后一种调用格式中，可以对多变量样条对象进行偏导数求取。

【例 8-14】考虑例 8-11 中给出的数据点，试用三次分段多项式样条函数与 B 样条插值函数求出该函数的导数，并与理论推导结果相比较。

【求解】可以用下面的语句生成原始数据，并分别建立起三次分段多项式型样条函数与 B 样条函数的数据类，这样就可以调用 fnder() 函数求出该函数的的导数，并得出如图 8-16 所示曲线。

```
>> syms x; f=(x^2-3*x+5)*exp(-5*x)*sin(x); ezplot(diff(f),[0,1]), hold on
   x=0:.12:1; y=(x.^2-3*x+5).*exp(-5*x).*sin(x);
   sp1=csapi(x,y); dsp1=fnder(sp1,1); fnplt(dsp1,'--')
   sp2=spapi(5,x,y); dsp2=fnder(sp2,1); fnplt(dsp2,':'); axis([0,1,-0.8,5])
```

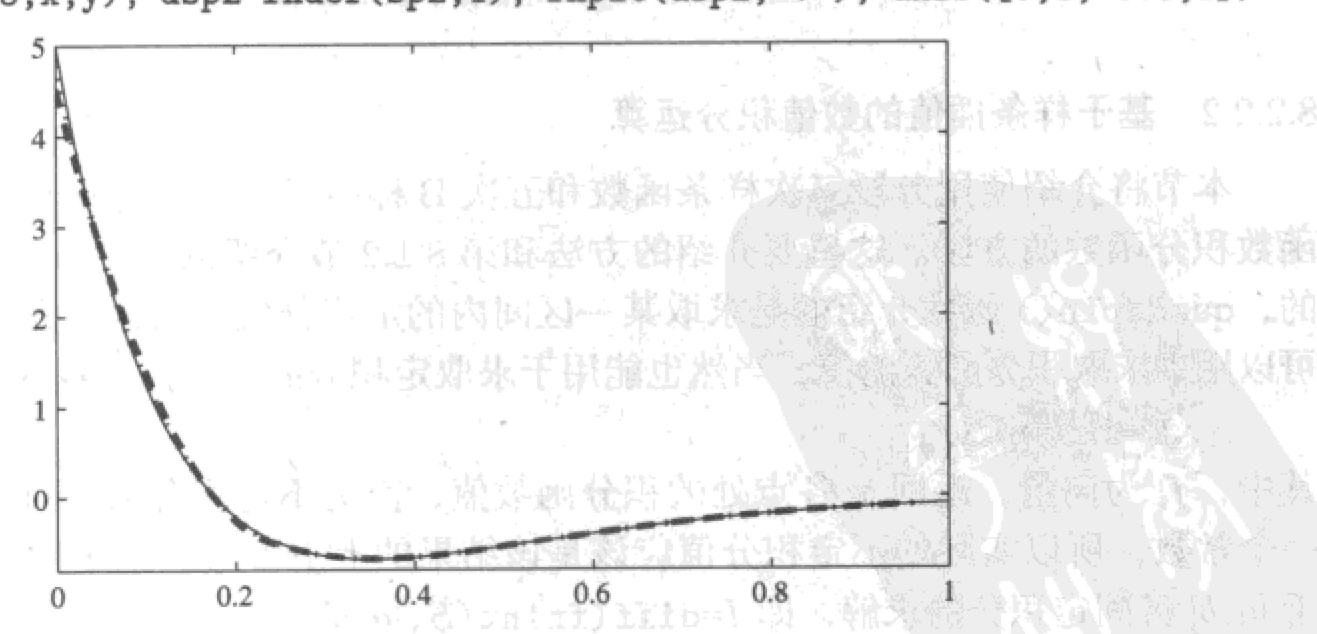


图 8-16 基于样条插值的函数数值微分结果

在图 8-16 中同时还绘制理论曲线。可见，用 B 样条拟合的数值微分结果是相当精确的，用三

次分段多项式样条插值算法的得出的微分效果也是很理想的，因为给出的数据点是很稀疏的，用第 3.3 节中给出的算法无法得出这样的结果。

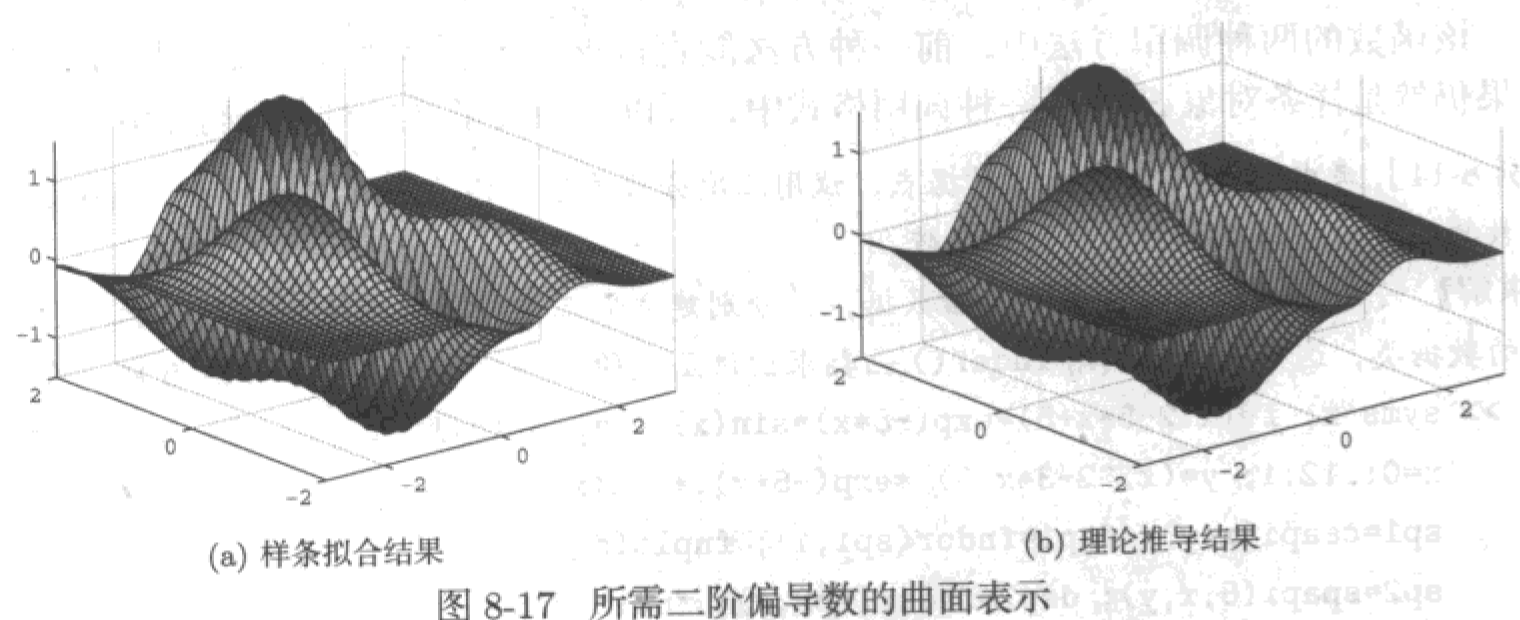
【例 8-15】 试由例 8-12 中给出的数据拟合 $\partial^2 z/(\partial x \partial y)$ 的曲面，并将得出的结果与解析解法绘制出的曲面相比较。

【求解】 由下面给出的语句可以直接生成数据，进行 B 样条函数拟合，并对得出的结果进行求导，绘制出如图 8-17 (a) 所示的曲面。

```
>> x0=-3:.3:3; y0=-2:.2:2; [x,y]=ndgrid(x0,y0);
   z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
   sp=spapi({5,5},{x0,y0},z); dspxy=fnder(sp,[1,1]); fnplt(dspxy)
```

当然，下面的语句可以用理论的方法推导出所需的偏导数，并绘制出其曲面，如图 8-17 (b) 所示。可见，这样得出的曲面与样条插值得出的结果完全一致。由此可以看出，基于样条插值的数值偏导数算法函数是可靠的。

```
>> syms x y; z=(x^2-2*x)*exp(-x^2-y^2-x*y);
   ezsurf(diff(diff(z,x),y),[-3 3],[-2 2])
```



(a) 样条拟合结果 (b) 理论推导结果

图 8-17 所需二阶偏导数的曲面表示

8.2.2.2 基于样条插值的数值积分运算

本节将介绍使用分段三次样条函数和五次 B 样条函数来逼近被积函数，从而求取某函数积分函数的方法。这里要介绍的方法和第 8.1.2 节介绍的 quadspln() 函数是有区别的，quadspln() 函数介绍的是求取某一区间内的定积分值，而这里介绍的 fnint() 方法可以用来求取积分函数的值，当然也能用于求取定积分的值。该函数的调用格式为

f_i=fnint(S)

其中，f_i 为向量，返回 x 各点处的积分函数值，因为不定积分通常可以在积分结果上加一个常数，所以实际的不定积分值应该是该结果的上下平移。该不定积分结果还可以用于 $[a,b]$ 区间定积分的求解，即 I=diff(fnint(S,$[a,b]$))。

【例 8-16】 仍考虑例 8-4 中较稀疏的样本点，试用样条积分的方式求出定积分及积分函数。

【求解】 下面的语句可以用两种形式建立起插值对象，用 fnint() 函数可以分别得出积分函数，

并求出所需的定积分值。可见，这样得出的结果远远比例 8-4 中得出的结果精确得多，用 B 样条甚至能在极稀疏的样本点前提下得出相当高精度的定积分结果。

```
>> x=[0,0.4,1 2,pi]; y=sin(x);
   sp1=csapi(x,y); a=fnint(sp1,1); xx=fnval(a,[0,pi]); xx(2)-xx(1)
ans =
   2.01905234585838
>> sp2=spapi(5,x,y); b=fnint(sp2,1); xx=fnval(b,[0,pi]); xx(2)-xx(1)
ans =
   1.99994177102332
```

用下面的语句还可以绘制出积分函数的曲线，其中由 B 样条函数可以得出和解析解十分接近的积分函数，如图 8-18 所示。

```
>> ezplot('-cos(t)+2',[0,pi]); hold on
   fnplt(a,'--'); fnplt(b,':')
```

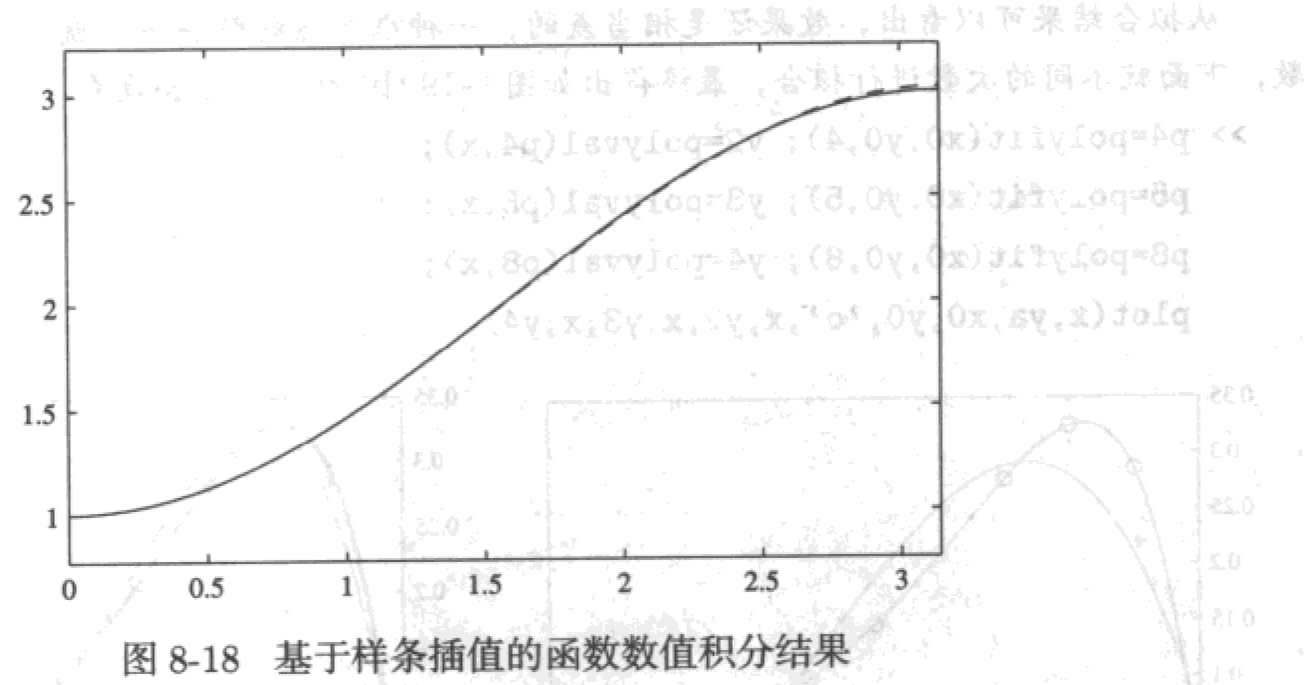


图 8-18 基于样条插值的函数数值积分结果

8.3 由已知数据拟合数学模型

8.3.1 多项式拟合

前面介绍的 Lagrange 拟合就是一种多项式拟合。一般多项式拟合的目标是找出一组多项式系数 $a_i, i=1,2,\cdots,n+1$，使得多项式

$$\psi(x)=a_1x^n+a_2x^{n-1}+\cdots+a_nx+a_{n+1} \tag{8-3-1}$$

能够较好地拟合原始数据。和前面介绍的插值算法不同，多项式拟合并不能保证每个样本点都在拟合的曲线上，但能使得整体的拟合误差较小。多项式拟合可以通过 MATLAB 提供的 polyfit() 函数实现。该函数的调用格式为

```
p=polyfit(x,y,n)
```

其中，$\boldsymbol{x}$ 和 $\boldsymbol{y}$ 为原始的样本点构成的向量，n 为选定的多项式阶次，得出的 $\boldsymbol{p}$ 为多项式系

数按降幂排列得出的行向量，可以用符号运算工具箱中的 poly2sym() 函数将其转换成真正的多项式形式，也可以使用 polyval() 函数求取多项式的值。下面将通过例子演示多项式拟合函数的使用方法和优缺点。

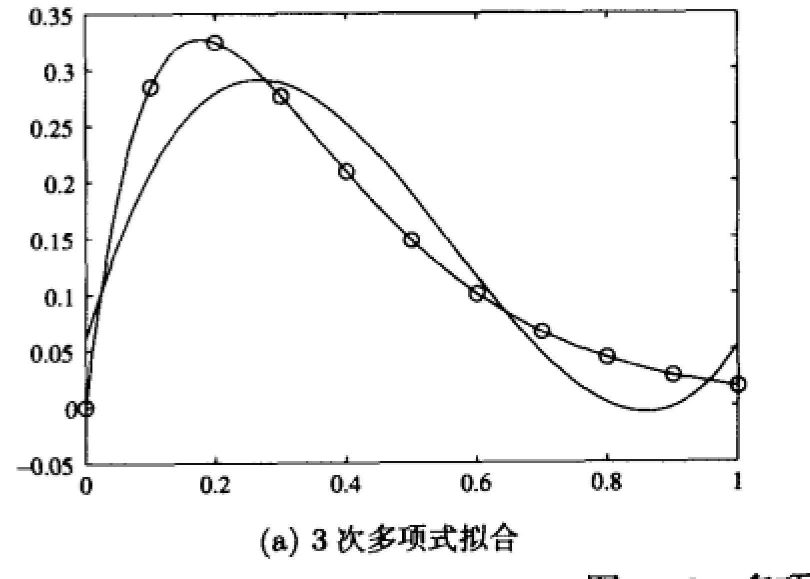
【例 8-17】考虑例 8-1 中的样本点数据，试用多项式拟合的方法在不同的阶次下进行拟合，并观察拟合效果，找出合适的阶次。

【求解】可以用下面语句得出拟合该数据的 3 次多项式并绘制出拟合曲线，如图 8-19 (a) 所示。

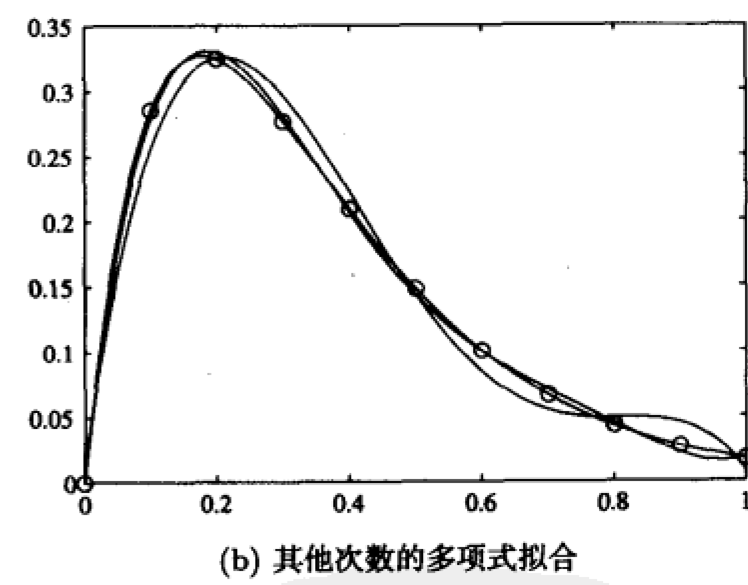
```
>> x0=0:.1:1; y0=(x0.^2-3*x0+5).*exp(-5*x0).*sin(x0);
   p3=polyfit(x0,y0,3); vpa(poly2sym(p3),10) % 可以如下显示多项式
ans =
   2.839962923*x^3-4.789842696*x^2+1.943211631*x+.5975248921e-1
>> x=0:.01:1; ya=(x.^2-3*x+5).*exp(-5*x).*sin(x);
   y1=polyval(p3,x); plot(x,y1,x,ya,x0,y0,'o')
```

从拟合结果可以看出，效果还是相当差的，一种很显然的解决方法就是增加拟合多项式的次数，下面就不同的次数进行拟合，最终得出如图 8-19 (b) 所示的拟合效果。

```
>> p4=polyfit(x0,y0,4); y2=polyval(p4,x);
   p5=polyfit(x0,y0,5); y3=polyval(p5,x);
   p8=polyfit(x0,y0,8); y4=polyval(p8,x);
   plot(x,ya,x0,y0,'o',x,y2,x,y3,x,y4)
```



(a) 3 次多项式拟合



(b) 其他次数的多项式拟合

图 8-19　多项式拟合效果

从该例的拟合效果看，$n \geqslant 8$ 就能得出较好的结果，这时拟合多项式可以由下面的语句显示出来，可以用该多项式去近似原函数模型。

```
>> vpa(poly2sym(p8),5)
ans =
   -8.2586*x^8+43.566*x^7-101.98*x^6+140.22*x^5-125.29*x^4+
      74.450*x^3-27.672*x^2+4.9869*x+.42037e-6
```

多项式拟合实际上相当于对已知函数用 Taylor 幂级数表示，但 Taylor 幂级数展开的前提条件是函数应该已知，这对实际的多项式拟合问题显得很苛刻。对本例来说，因为原函数是已知

的，所以可以通过 Taylor 幂级数方法先展开该函数，可以得出如下的结果：

```
>> syms x; y=(x^2-3*x+5)*exp(-5*x)*sin(x);
   vpa(taylor(y,9),5)
ans =
 5.*x-28.*x^2+77.667*x^3-142.*x^4+192.17*x^5-204.96*x^6+179.13*x^7-131.67*x^8
```

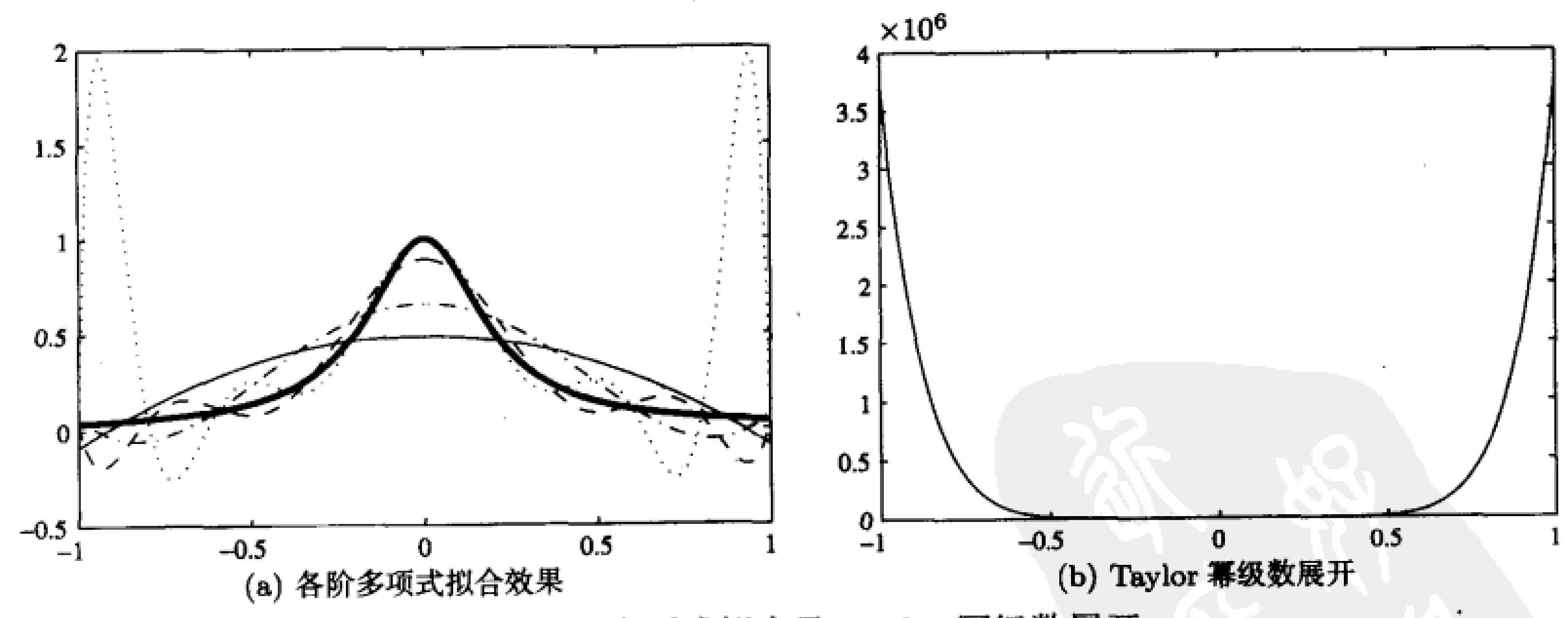
比较该结果和上述的多项式拟合的结果，可以发现二者是完全不同的，这样就可以得出结论，用多项式表示数据的模型是不惟一的，即使两个多项式函数完全不同。在某一区域内其曲线将特别近似。所以有时多项式拟合时应该注意检验结果，比如得出的结果是否很平滑，而不应片面地比较多项式的系数是否一致。

【例 8-18】 重新考虑例 8-3 中的函数，试观察多项式拟合的效果。

【求解】 多项式拟合的效果并不一定总是很精确的。考虑例 8-3 中的样本点，可以取不同的多项式阶次 n，则使用如下语句获得多项式拟合，并绘制出拟合曲线，如图 8-20 (a) 所示。

```
>> x0=-1+2*[0:10]/10; y0=1./(1+25*x0.^2);
   x=-1:.01:1; ya=1./(1+25*x.^2);
   p3=polyfit(x0,y0,3); y1=polyval(p3,x);
   p5=polyfit(x0,y0,5); y2=polyval(p5,x);
   p8=polyfit(x0,y0,8); y3=polyval(p8,x);
   p10=polyfit(x0,y0,10); y4=polyval(p10,x);
   plot(x,ya,x,y1,x,y2,'-.',x,y3,'--',x,y4,':')
```

其实，该例子如果用 Taylor 幂级数展开效果将更差，用下面的语句可以得出 Taylor 幂级数展开式及拟合效果，并可以绘制出该多项式拟合的效果，如图 8-20 (b) 所示。可以看出，这样拟合的结果是相当差的，甚至说是完全错误的。



(a) 各阶多项式拟合效果　　(b) Taylor 幂级数展开

图 8-20 多项式拟合及 Taylor 幂级数展开

```
>> syms x; y=1/(1+25*x^2); p=taylor(y,x,10)
p =
  1-25*x^2+625*x^4-15625*x^6+390625*x^8
>> x1=-1:0.01:1; ya=1./(1+25*x1.^2); y1=subs(p,x,x1); plot(x1,y1)
```

8.3.2　给定函数的连分式展开及基于连分式的有理近似

连分式是对函数或数值的一种很有效的近似形式。函数 $f(x)$ 经常可以用连分式形式表示为

$$f(x)=\cfrac{1}{f_1(x)+\cfrac{1}{f_2(x)+\cfrac{1}{f_3(x)+\cfrac{1}{f_4(x)+\cfrac{1}{\cdots}}}}} \tag{8-3-2}$$

而最常见的连分式形式为 Cauer II 型连分式[51]，表示为

$$f(x)=\cfrac{\alpha_1}{\beta_1+\cfrac{\alpha_2 x}{\beta_2+\cfrac{\alpha_3 x}{\beta_3+\cfrac{\alpha_4 x}{\beta_4+\cfrac{\alpha_5 x}{\cdots}}}}} \tag{8-3-3}$$

MATLAB 语言及符号运算工具箱并未直接提供连分式展开的函数，但可以调用 Maple 中给出的 `cfrac()` 函数来求取函数的连分式展开，在调用该函数前还需要将 Maple 的数论包用 `with()` 函数调入，这样给定函数或数值的 Cauer II 型连分式展开可以用下面的命令实现。

```
maple('with(numtheory):')                    调入数论包
f=maple(['cfe:=cfrac(' fun ',x,n)']);        调用连分式函数，生成 cfe 变量
```

其中，该函数将 MATLAB 定义的函数字符串 `fun` 进行连分式展开，自变量为 x，展开的项数为 n，该函数得出的部分分式展开 `cfe`，为 Maple 环境中的变量，而 f 为返回到 MATLAB 环境中的字符串。若对数值进行连分式展开，则可以不给出 x 变量。

由保留的前 n 级连分式的项，可以调用 Maple 语言的 `thenumer()` 和 `thedenom()` 函数变换出有理函数的近似形式。这两个函数的调用格式为

```
p=maple('nthnumer','cfe',n);    由 cfe 变量提取前 n 级的分子
q=maple('nthdenom','cfe',n);    由 cfe 变量提取前 n 级的分母
```

由上面两个命令可以得出有理近似的分子和分母。

【例 8-19】先观察一个常数的连分式近似问题，试对 π 进行 20 级近似，并找出一个较好的连分式近似阶次。

【求解】一个常数的连分式可以用下面的语句直接得出：

```
>> maple('with(numtheory):'); f=maple(['cfe:=cfrac(pi,20)'])
f =
```

```
cfe := 3+1/(7+1/(15+1/(1+1/(292+1/(1+1/(1+1/(1+1/(2+1/(1+1/(3+1/(1+
   1/(14+1/(2+1/(1+1/(1+1/(2+1/(2+1/(2+1/(2+1/(1+`...`)))))))))))))))))))))
```

亦即 π 的连分式展开式子为

$$\pi \simeq 3+\cfrac{1}{7+\cfrac{1}{15+\cfrac{1}{1+\cfrac{1}{292+\cfrac{1}{1+\cdots}}}}}$$

其中，分子 292 和其他值相比差得较悬殊，所以截断到此级即可以得出较高的精度。由有理近似的函数则可以得出分子和分母的值为

```
>> n=maple('nthnumer','cfe',4); d=maple('nthdenom','cfe',4); [vpa(n),vpa(d)]
ans =
    [ 103993.,  33102.]
```

这时还可以得出 4 级连分式有理近似为 $\dfrac{103993}{33102} \simeq 3.14159265301190260407226149477 37$。可见，只用 4 级连分式近似就相当接近 π 值了。

【例 8-20】 试用连分式展开的方法求出函数 $f(x)=\dfrac{\sin(x)\mathrm{e}^{-x}}{(x+1)^3}$ 的前 10 级表达式，并求出该函数的有理近似表达式。

【求解】 根据要求，可以用下面的语句立即得出前 10 级连分式表达式。

```
>> syms x; fun='sin(x)*exp(-x)/(x+1)^3';  % fun 应该为字符串
   maple('with(numtheory):'); f=maple(['cfe:=cfrac(' fun ',x,10)'])
f =
   cfe := x/(1+4*x/(1-5*x/(3+43*x/(20-337*x/(43+28274*x/(1685-66157779*x/
          (395836-9881300005*x/(512851+140501598188444*x/(158335371-
          531240292464601408*x/(2484643103+`...`))))))))))
```

亦即这样可以得出

$$f(x)=\cfrac{x}{1+\cfrac{4x}{1-\cfrac{5x}{3+\cfrac{43x}{20-\cfrac{337x}{43+\cfrac{28274x}{1685-\cfrac{66157779x}{395836-\cfrac{9881300005x}{512851+\cfrac{140501598188444}{158335371-\cfrac{531240292464601408}{2484643103+\cdots}}}}}}}}}}$$

由下面的语句可以得出前 8 级和 10 级连分式的有理多项式近似。

```
>> n=collect(maple('nthnumer','cfe',8),x);  % 分子多项式合并同类项
   d=collect(maple('nthdenom','cfe',8),x); [n,d]=numden(n/d); G=n/d; latex(G)
   n=collect(maple('nthnumer','cfe',10),x);
   d=collect(maple('nthdenom','cfe',10),x);
   [n,d]=numden(n/d); G1=n/d; latex(G1) % 显示从略
```

这时可以得出

$$f_8(x) \simeq 10\frac{x\left(845713x^3 - 4973560x^2 + 11841438x - 10769871\right)}{5864273x^4 - 83147900x^3 - 294069480x^2 - 312380460x - 107698710}$$

用下面的语句还可以得出 $(0, 0.5)$ 区间内的原始函数 $f(x)$ 和 $f_8(x)$ 的曲线，如图 8-21 (a) 所示。可见，拟合效果还是很理想的，$n = 10$ 时效果更好些，几乎无法区分原函数曲线和拟合曲线。若扩大拟合区域，令其为 $(0, 5)$，则可以得出如图 8-21 (b) 所示的拟合曲线，可见这样的拟合效果变差，需要进一步增加连分式级数，所以这样的方法有时不适合于大区域拟合。

```
>> ezplot(fun,[0,2]), hold on; ezplot(G,[0,2]); ezplot(G1,[0,2])
   figure;  ezplot(fun,[0,5]), hold on; ezplot(G,[0,5]); ezplot(G1,[0,5])
```

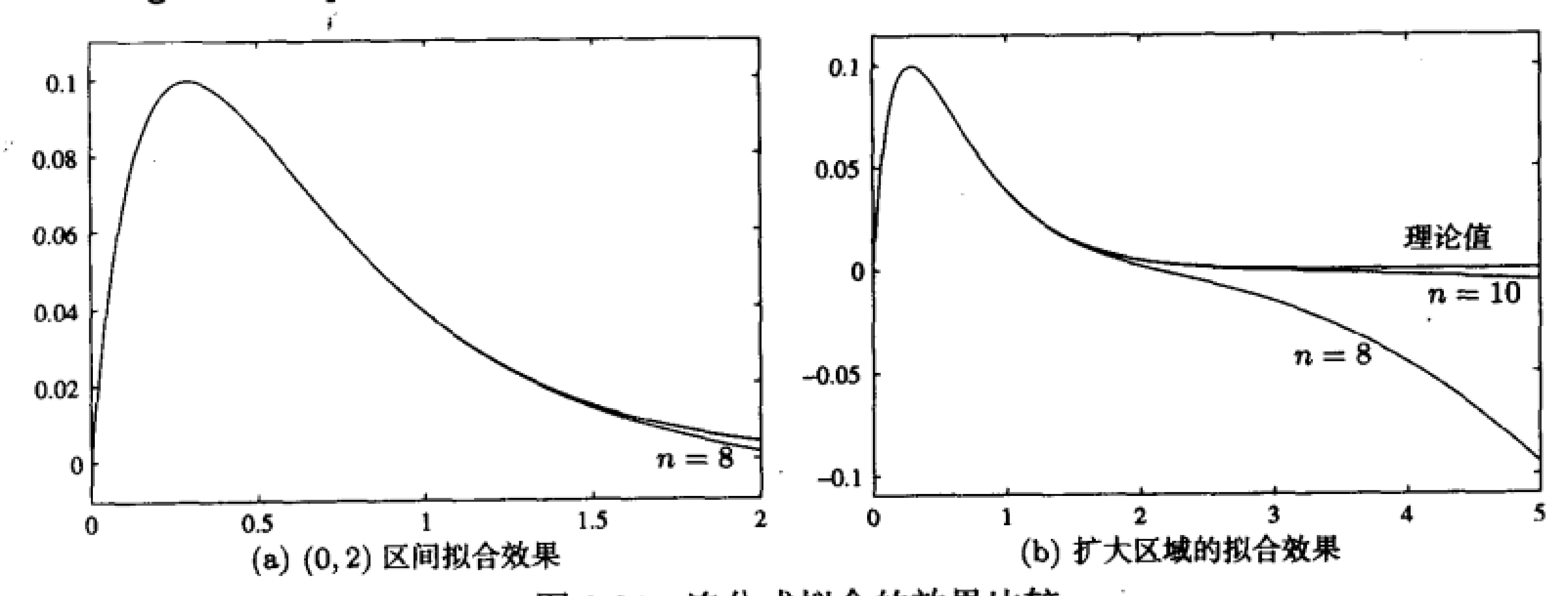


图 8-21 连分式拟合的效果比较

8.3.3 有理式拟合 — Padé 近似

假设某函数 $f(s)$ 的幂级数展开可以表示为

$$f(s) = c_0 + c_1 s + c_2 s^2 + c_3 s^3 + \cdots = \sum_{i=0}^{\infty} c_i s^i \tag{8-3-4}$$

并假设 r/m 阶的 Padé 近似可以写成如下的有理函数形式

$$G_m^r(s) = \frac{\beta_{r+1}s^r + \beta_r s^{r-1} + \cdots + \beta_1}{\alpha_{m+1}s^m + \alpha_m s^{m-1} + \cdots + \alpha_1} = \frac{\displaystyle\sum_{i=1}^{r+1} \beta_i s^{i-1}}{\displaystyle\sum_{i=1}^{m+1} \alpha_i s^{i-1}} \tag{8-3-5}$$

式中，$\alpha_1=1$, $\beta_1=c_1$。设 $\sum_{i=0}^{\infty}c_is^i=G_m^r(s)$，则可以写出如下的等式

$$\sum_{i=1}^{m+1}\alpha_is^{i-1}\sum_{i=0}^{\infty}c_is^i=\sum_{i=1}^{r+1}\beta_is^{i-1} \tag{8-3-6}$$

对比等式中 s 相应次数的系数，令相应的 s 项系数的值相等，则 α_i, $i=2,\cdots,m+1$ 和 β_i, $i=2,\cdots,k+1$ 系数可以从下面的方程求解出来。

$$\boldsymbol{W}\boldsymbol{x}=\boldsymbol{w},\quad \boldsymbol{v}=\boldsymbol{V}\boldsymbol{y} \tag{8-3-7}$$

其中，

$$\begin{aligned}\boldsymbol{x}&=[\alpha_2,\alpha_3,\cdots,\alpha_{m+1}]^{\mathrm{T}},\ \boldsymbol{w}=[-c_{r+2},-c_{r+3},\cdots,-c_{m+r+1}]^{\mathrm{T}}\\ \boldsymbol{v}&=[\beta_2-c_2,\beta_3-c_3,\cdots,\beta_{r+1}-c_{r+1}]^{\mathrm{T}},\ \boldsymbol{y}=[\alpha_2,\alpha_3,\cdots,\alpha_{r+1}]^{\mathrm{T}}\end{aligned} \tag{8-3-8}$$

且

$$\boldsymbol{W}=\begin{bmatrix}c_{r+1}&c_r&\cdots&0&\cdots&0\\c_{r+2}&c_{r+1}&\cdots&c_1&\cdots&0\\\vdots&\vdots&\cdots&\vdots&\ddots&\vdots\\c_{r+m}&c_{r+m-1}&\cdots&c_{m-1}&\cdots&c_{r+1}\end{bmatrix} \tag{8-3-9}$$

$$\boldsymbol{V}=\begin{bmatrix}c_1&0&0&\cdots&0\\c_2&c_1&0&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\c_r&c_{r-1}&c_{r-2}&\cdots&c_1\end{bmatrix} \tag{8-3-10}$$

可以证明[2]，若 Padé 近似的分子分母阶次相同或分母比分子高一阶，则该近似等效于 Cauer II 型连分式近似。可以编写一个 MATLAB 函数 padefcn() 来计算给定 $f(x)$ 函数的 Padé 有理函数近似。该函数的清单如下：

```
function [nP,dP]=padefcn(c,r,m)
w=-c(r+2:m+r+1)'; vv=[c(r+1:-1:1)'; zeros(m-1-r,1)];
W=rot90(hankel(c(m+r:-1:r+1),vv)); V=rot90(hankel(c(r:-1:1)));
x=[1 (W\w)']; y=[1 x(2:r+1)*V'+c(2:r+1)];
dP=x(m+1:-1:1)/x(m+1); nP=y(r+1:-1:1)/x(m+1);
```

【例 8-21】试对 $f(x)=\mathrm{e}^{-2x}$ 函数用有理函数近似。

【求解】可以选择不同的分母阶次，选择分子阶次为 0，并选择不同的分母阶次，则可以得出不同的 Padé 有理近似式，近似曲线如图 8-22 所示。

```
>> syms x; c=taylor(exp(-2*x),10); c=sym2poly(c); c=c(end:-1:1); x=0:0.01:8;
   nd=[3:7]; xx=[0,2,2+eps,8]; yy=[0,0,1,1]; plot(xx,yy); hold on
   for i=1:length(nd)
      [n,d]=padefcn(c,0,nd(i)); y=polyval(n,x)./polyval(d,x); plot(x,y)
   end
```

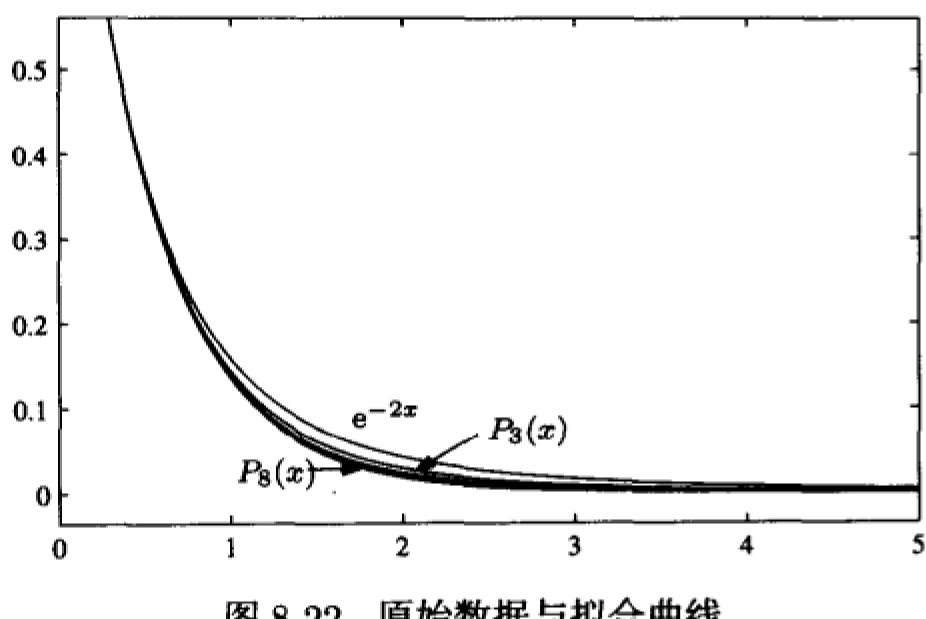


图 8-22 原始数据与拟合曲线

由图 8-22 可见，3 阶近似得出的效果尚可，如果增加阶次，会得出更好的效果，8 阶近似的结果还是很精确的。8 阶 Padé 近似表达式如下：

$$P_8(s)=\frac{157.5}{x^8+4x^7+14x^6+42x^5+105x^4+210x^3+315x^2+315x+157.5}$$

8.3.4 函数线性组合的曲线拟合方法

假设已知某函数的线性组合为

$$g(x)=c_1f_1(x)+c_2f_2(x)+c_3f_3(x)+\cdots+c_nf_n(x) \tag{8-3-11}$$

其中，$f_1(x),f_2(x),\cdots,f_n(x)$ 为已知函数，$c_1,c_2,\cdots,c_n$ 为待定系数，这时假设已经测出数据 $(x_1,y_1),(x_2,y_2),\cdots,(x_M,y_M)$，则可以建立起如下的线性方程

$$\boldsymbol{A}\boldsymbol{c}=\boldsymbol{y} \tag{8-3-12}$$

其中，

$$\boldsymbol{A}=\begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1)\\ f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2)\\ \vdots & \vdots & \ddots & \vdots\\ f_1(x_M) & f_2(x_M) & \cdots & f_m(x_M)\end{bmatrix},\quad \boldsymbol{y}=\begin{bmatrix} y_1\\ y_2\\ \vdots\\ y_M\end{bmatrix} \tag{8-3-13}$$

且 $\boldsymbol{c}=[c_1,c_2,\cdots,c_n]^{\mathrm{T}}$。故该方程的最小二乘解为 $\boldsymbol{c}=\boldsymbol{A}\backslash\boldsymbol{y}$。

【例 8-22】假设测出了一组 (x_i,y_i)，由下面的表格给出，且已知函数原型为 $y(x)=c_1+c_2\mathrm{e}^{-3x}+c_3\cos(-2x)\mathrm{e}^{-4x}+c_4x^2$，试用已知数据求出待定系数 c_i 的值。

x_i	0	0.2	0.4	0.7	0.9	0.92	0.99	1.2	1.4	1.48	1.5
y_i	2.88	2.2576	1.9683	1.9258	2.0862	2.109	2.1979	2.5409	2.9627	3.155	3.2052

【求解】可以将表中数据直接拟合出曲线方程中的 c_i 参数。

```
>> x=[0,0.2,0.4,0.7,0.9,0.92,0.99,1.2,1.4,1.48,1.5]';
   y=[2.88;2.2576;1.9683;1.9258;2.0862;2.109;2.1979;2.5409;2.9627;3.155;3.2052];
   A=[ones(size(x)) exp(-3*x), cos(-2*x).*exp(-4*x) x.^2];
   c=A\y; c1=c'
c' =
    1.22002081340577    2.33972067465903   -0.67973291882833    0.86999835216114
>> x0=[0:0.01:1.5]';
   A1=[ones(size(x0)) exp(-3*x0), cos(-2*x0).*exp(-4*x0) x0.^2];
   y1=A1*c; plot(x0,y1,x,y,'x')
```

这时可以得出拟合曲线和已知数据点，如图 8-23 所示。可见拟合效果是令人满意的。

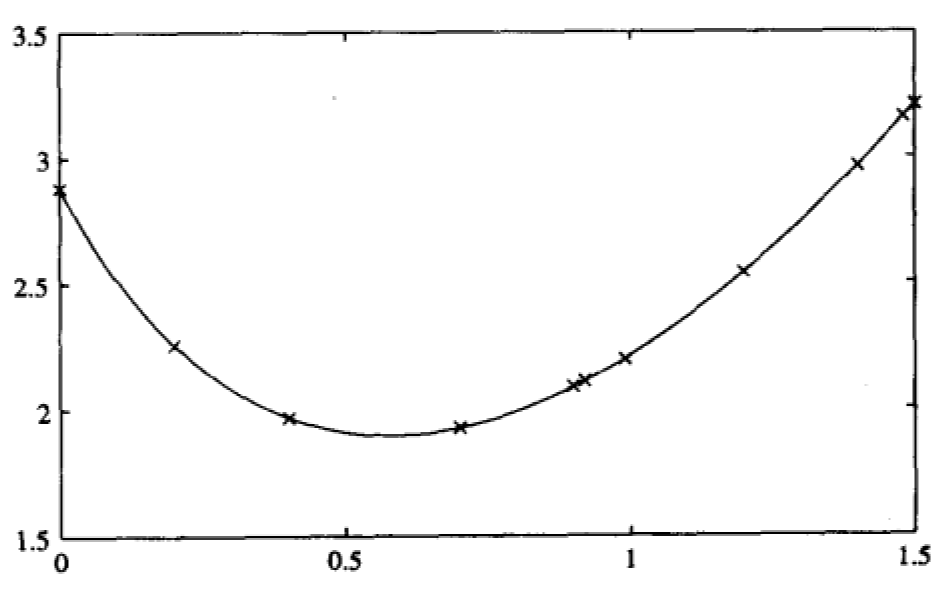


图 8-23 原始数据与拟合曲线

【例 8-23】假设测出一组实际数据，试对其进行函数拟合。

x_i	1.1052	1.2214	1.3499	1.4918	1.6487	1.8221	2.0138	2.2255	2.4596	2.7183	3.6693
y_i	0.6795	0.6006	0.5309	0.4693	0.4148	0.3666	0.3241	0.2865	0.2532	0.2238	0.1546

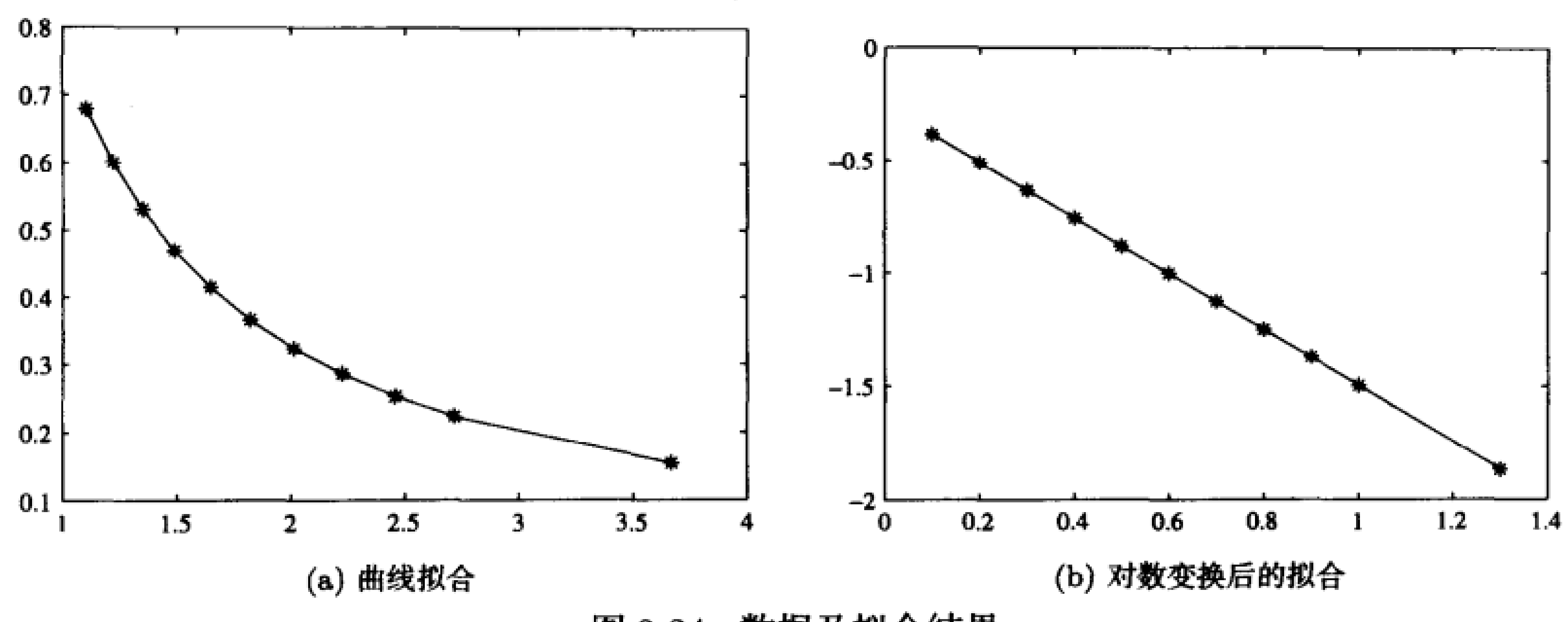
【求解】可以用下面的语句将表中给出的数据用曲线表示出来，如图 8-24 (a) 所示。

```
>> x=[1.1052,1.2214,1.3499,1.4918,1.6487,1.8221,2.0138,...
      2.2255,2.4596,2.7183,3.6693];
   y=[0.6795,0.6006,0.5309,0.4693,0.4148,0.3666,0.3241,...
      0.2864,0.2532,0.2238,0.1546];
   plot(x,y,x,y,'*')
```

在实际曲线拟合时，有时从 x,y 本身看不出它们之间的关系，则可能需要对数据进行可能的非线性变换，观察是否能得出线性关系。例如，可以对 x,y 分别进行对数变换，得出如图 8-24 (b) 所示的曲线，可见二者是线性的。

```
>> x1=log(x); y1=log(y);  plot(x1,y1,x1,y1,'*')
```

这样用线性函数拟合的方法则可以得出线性参数，使得 $\ln y = a\ln x + b$，亦即 $y = e^b x^a$，而系数 a,b 及 e^b 可以由下面的语句直接得出。



(a) 曲线拟合　　(b) 对数变换后的拟合

图 8-24 数据及拟合结果

```
>> A=[x1' ones(size(x1'))]; c=[A\y1']'
c =
  -1.23389448522593        -0.26303708610220
>> exp(c(2))
ans =
    0.76871338819924
```

亦即可以得出拟合函数 $y(x) = 0.76871338819924x^{-1.23389448522593}$。

【例 8-24】 多项式拟合可以认为是前面介绍的多函数线性组合的特例，这样可以选择各个函数为 $f_i(x) = x^{n+1-i}$，$i = 1, 2, \cdots, n$，用该方法重新考虑例 8-17 中数据的多项式拟合问题，试观察多项式拟合的效果。

【求解】 由上述的算法，可以立即得出数据的多项式拟合结果，该结果和例 8-17 给出的结果完全一致。

```
>> x=[0:.1:2]'; y=(x.^2-3*x+5).*exp(-5*x).*sin(x); n=7; A=[];
   for i=1:n+1, A(:,i)=x.^(n+1-i); end
   c=A\y; vpa(poly2sym(c),5)
ans =
     -8.2586*x^8+43.566*x^7-101.98*x^6+140.22*x^5-125.29*x^4+
      74.450*x^3-27.672*x^2+4.9869*x+.42037e-6
```

8.3.5 最小二乘曲线拟合

假设有一组数据 $x_i, y_i, i = 1, 2, \cdots, N$，且已知这组数据满足某一函数原型 $\hat{y}(x) = f(\boldsymbol{a}, x)$，其中 $\boldsymbol{a}$ 待定系数向量，则最小二乘曲线拟合的目标就是求出这一组待定系数的值，使得目标函数

$$J = \min_{\boldsymbol{a}} \sum_{i=1}^{N} [y_i - \hat{y}(x_i)]^2 = \min_{\boldsymbol{a}} \sum_{i=1}^{N} [y_i - f(\boldsymbol{a}, x_i)]^2 \tag{8-3-14}$$

为最小。在 MATLAB 的最优化工具箱中提供了 lsqcurvefit() 函数，可以解决最小二乘曲线拟合的问题。该函数的调用格式为

$[\boldsymbol{a},J_m]$=lsqcurvefit(Fun,$\boldsymbol{a}_0$,$\boldsymbol{x}$,$\boldsymbol{y}$)

其中，Fun 为原型函数的 MATLAB 表示，可以是 M-函数或 inline() 函数，$\boldsymbol{a}_0$ 为最优化的初值，$\boldsymbol{x}$, $\boldsymbol{y}$ 为原始输入输出数据向量，调用该函数则将返回待定系数向量 $\boldsymbol{a}$ 以及在此待定系数下的目标函数的值 J_m。

【例 8-25】假设由下面的语句生成一组数据 x 和 y

```
>> x=0:.1:10; y=0.12*exp(-0.213*x)+0.54*exp(-0.17*x).*sin(1.23*x);
```

并已知该数据满足原型为 $y(x)=a_1\mathrm{e}^{-a_2x}+a_3\mathrm{e}^{-a_4x}\sin(a_5x)$，其中，$a_i$ 为待定系数。采用最小二乘曲线拟合的目的就是获得这些待定系数，使得目标函数的值为最小。

【求解】根据已知的函数原型，可以编写出如下的 MATLAB 函数：

```
>> f=inline('a(1)*exp(-a(2)*x)+a(3)*exp(-a(4)*x).*sin(a(5)*x)','a','x');
```

建立起函数的原型，则可以由下面的语句得出待定系数向量了。

```
>> [xx,res]=lsqcurvefit(f,[1,1,1,1,1],x,y); xx',res
Optimization terminated successfully:
 Relative function value changing by less than OPTIONS.TolFun
ans =                        res =
   0.11971712065965             7.163683226738623e-007
   0.21246047792014
   0.54040192789832
   0.17018571958912
   1.23002540347539
```

可以看出，这样得出的待定系数精度较高，接近与理论值 $\boldsymbol{a}=[0.12,0.213,0.54,0.17,1.23]^{\mathrm{T}}$。如果想进一步提高精度，则需要修改最优化的选项，这时函数的调用格式也将发生变化。

```
>> ff=optimset; ff.TolFun=1e-20; ff.TolX=1e-15; % 修改精度限制
   [xx,res]=lsqcurvefit(f,[1,1,1,1,1],x,y,[],[],ff); xx',res
Optimization terminated successfully:
 Relative function value changing by less than OPTIONS.TolFun
ans =                        res =
   0.11999999996745             9.503499130199283e-021
   0.21299999993794
   0.54000000004622
   0.17000000002139
   1.23000000000295
>> x1=0:0.01:10; y1=f(xx,x1); plot(x1,y1,x,y,'o')
```

其中，两个空矩阵表示 $\boldsymbol{a}$ 向量的上下限，由于对这些参数的范围无限制，故采用了默认的表示形式。可以看出，修改误差限后，得出的拟合待定系数更加精确。绘制出的拟合曲线与样本点如图 8-25 所示。

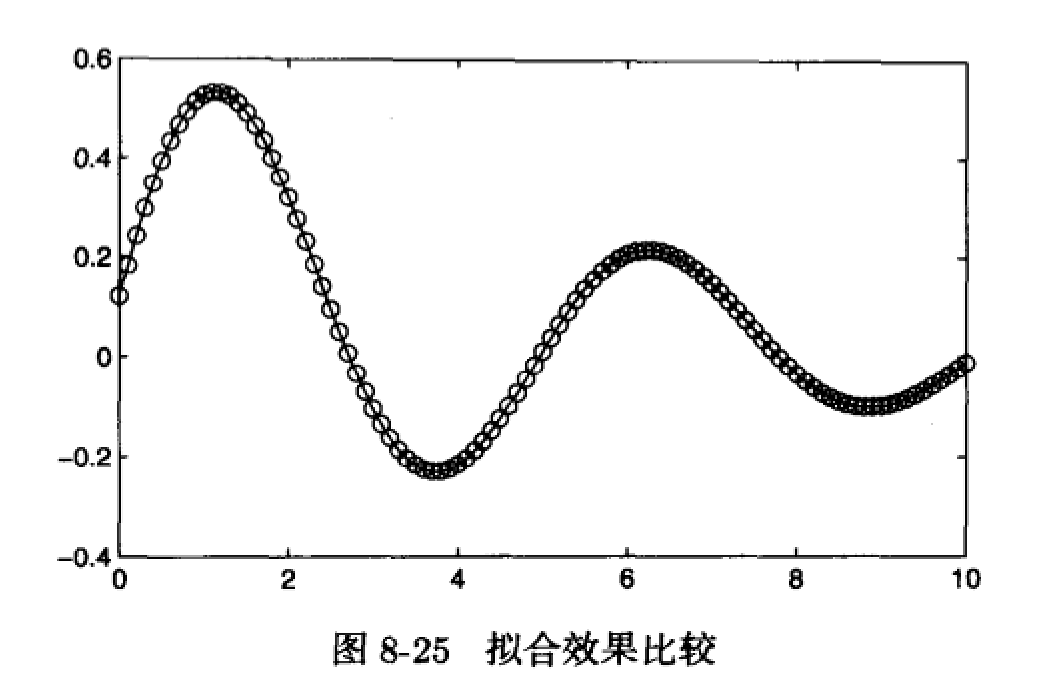


图 8-25 拟合效果比较

【例 8-26】假设有一组实测数据

x_i	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y_i	2.3201	2.6470	2.9707	3.2885	3.6008	3.9090	4.2147	4.5191	4.8232	5.1275

假设已知该数据可能满足的原型函数为 $y(x) = ax + bx^2\mathrm{e}^{-cx} + d$，试求出满足下面数据的最小二乘解 a, b, c, d 的值。

【求解】下面的语句可以输入已知的参数。

```
>> x=0.1:0.1:1;
   y=[2.3201,2.6470,2.9707,3.2885,3.6008,3.9090,4.2147,4.5191,4.8232,5.1275];
```

令 $a_1 = a, a_2 = b, a_3 = c, a_4 = d$，这样，原型函数可以写成 $y(x) = a_1x + a_2x^2\mathrm{e}^{-a_3x} + a_4$，可以用 MATLAB 写出

```
function y=c8f3(a,x)
y=a(1)*x+a(2)*x.^2 .*exp(-a(3)*x)+a(4);
```

则

```
>> a=lsqcurvefit('c8f3',[1;2;2;3],x,y); a'
ans =
   2.45871058396841   2.44894584435743   1.44655649377552   2.07337742284199
```

用下面的语句还可以计算出各个点处的值，可以将二者曲线绘制在同一坐标系下，如图 8-26 所示。可见，二者还是很接近的，说明拟合效果较好。

```
>> y1=c8f3(a,x); plot(x,y,x,y1,'o')
```

8.4 信号分析与数字信号处理基础

8.4.1 信号的相关分析

假设已知某确定性信号为 $x(t)$，则其自相关函数的定义为

$$R_{xx}(\tau) = \lim_{T\to\infty}\frac{1}{T}\int_0^T x(t)x(t+\tau)\mathrm{d}t,\ \ \tau \geqslant 0 \tag{8-4-1}$$

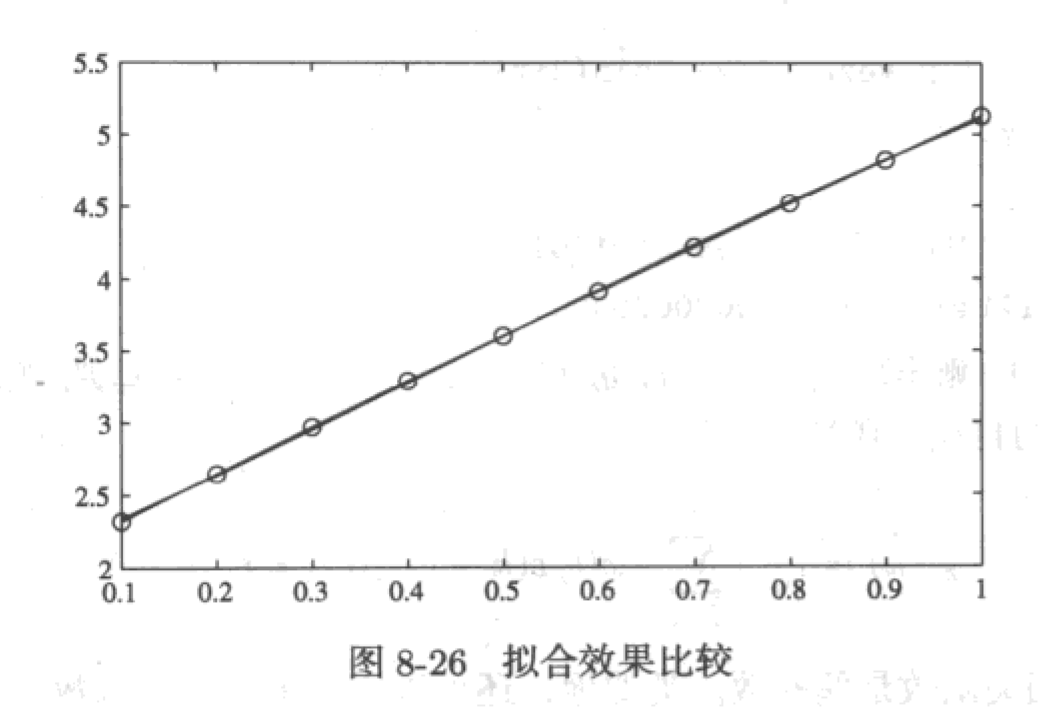


图 8-26 拟合效果比较

且相关函数为偶函数，即 $R_{xx}(-\tau)=R_{xx}(\tau)$。若研究两个信号 $x(t)$ 和 $y(t)$，则可以定义其互相关函数为

$$R_{xy}(\tau)=\lim_{T\to\infty}\frac{1}{T}\int_0^T x(t)y(t+\tau)\mathrm{d}t,\ \ \tau\geqslant 0 \tag{8-4-2}$$

从前面介绍的微积分求解方法可知，若这些函数都已知，则可以通过 MATLAB 的符号运算工具箱直接求出自相关函数和互相关函数的解析表达式。

【例 8-27】假设已知函数 $x(t)=A_1\cos(\omega_1 t+\theta_1)+A_2\cos(\omega_2 t+\theta_2)$，试根据定义求出该函数的自相关函数。

【求解】根据定义，首先应该申明一些有关符号变量，再定义出已知函数 $x(t)$，这样，就可以由下面的语句求出信号的自相关函数。

```
>> syms A1 A2 w1 w2 t1 t2 t tau T; x=A1*cos(w1*t+t1)+A2*cos(w2*t+t2);
   Rxx=simple(limit(int(x*subs(x,t,t+tau),t,0,T)/T,T,inf))
```

这样可以求出信号的自相关函数解析解为

$$R_{xx}(\tau)=\frac{1}{2}A_1^2\cos(\omega_1\tau)+\frac{1}{2}A_2^2\cos(\omega_2\tau)$$

假设在实验中测出两组数据，$x_i, y_i, (i=1,2,\cdots,n)$，则可以由下面的式子计算出两组数据的相关系数为

$$r_{xy}=\frac{\sqrt{\sum(x_i-\bar{x})(y_i-\bar{y})}}{\sqrt{\sum(x_i-\bar{x})^2}\sqrt{\sum(y_i-\bar{y})^2}} \tag{8-4-3}$$

MATLAB 提供了`corrcoef()` 函数，可以求出已知 $\boldsymbol{x},\boldsymbol{y}$ 向量的相关系数矩阵 $\boldsymbol{R}$。该函数的调用格式为

$\boldsymbol{R}$=corrcoef($\boldsymbol{x}$,$\boldsymbol{y}$) 或 **$\boldsymbol{R}$=corrcoef([$\boldsymbol{x}$,$\boldsymbol{y}$])**

【例 8-28】试用原型函数 $y_1=t\mathrm{e}^{-4t}\sin 3t$ 和 $y_2=t\mathrm{e}^{-4t}\cos 3t$ 分别生成一组数据，并求取其相关系数矩阵。

【求解】用下面的语句可以立即得出两个信号的相关系数矩阵为

```
>> x=0:0.01:5; y1=x.*exp(-4*x).*sin(3*x); y2=x.*exp(-4*x).*cos(3*x);
   R=corrcoef(y1,y2)
R =
   1.00000000000000   0.47758585121404
   0.47758585121404   1.00000000000000
```

仍假设在实验中测出两组数据，$x_i, y_i, (i=1,2,\cdots,n)$，对这些离散点可以由下面的式子定义 x_i 序列的自相关函数

$$c_{xx}(k)=\frac{1}{N}\sum_{l=1}^{n-[k]-1}x(l)x(k+l),\ 0\leqslant k\leqslant m-1 \tag{8-4-4}$$

其中 $m<n$，自相关函数是偶函数。类似地，还可以定义出互相关函数

$$c_{xy}(k)=\frac{1}{N}\sum_{l=1}^{n-[k]-1}x(l)y(k+l),\ 0\leqslant k\leqslant m-1 \tag{8-4-5}$$

相关函数可以用来研究两个序列信号的相似性。MATLAB 提供了求取和绘制自相关函数和互相关函数的程序，可以由 xcorr() 函数直接求解。该函数的调用格式为

C_{xx}=xcorr($\boldsymbol{x}$,N); C_{xy}=xcorr($\boldsymbol{x}$,$\boldsymbol{y}$,N)

其中，N 为 k 的最大取值，可以忽略。

【例 8-29】仍假设已知由例 8-28 中函数计算出的数据，试用数值方法求取自相关函数、互相关函数，并和已知理论曲线进行比较。

【求解】先在 $t\in(0,5)$ 区间生成一个时间向量，则可以由原型函数直接计算出 $\boldsymbol{x}$ 向量和 $\boldsymbol{y}$ 向量，调用现成的函数就可以得出它们的自相关函数和互相关函数，如图 8-27 (a)、图 8-27 (b) 所示。

```
>> t=0:0.01:5; x=t.*exp(-4*t).*sin(3*t); y=t.*exp(-4*t).*cos(3*t);
   N=150; c1=xcorr(x,N); x1=[-N:N]; stem(x1,c1)  % 实际绘图点选得更稀疏
   figure; c1=xcorr(x,y,N); x1=[-N:N]; stem(x1,c1)
```


(a) $\boldsymbol{x}$ 的自相关函数

(b) $\boldsymbol{x}$ 与 $\boldsymbol{y}$ 的互相关函数

图 8-27 信号的相关函数分析

8.4.2 快速 Fourier 变换

离散数据 $x_i, i=1,2,\cdots,N$ 的 Fourier 变换是数字信号处理的基础。离散 Fourier 变换的数学表示为

$$X(k)=\sum_{i=1}^{N} x_i \mathrm{e}^{-2\pi \mathrm{j}(k-1)(i-1)/N}, \text{ 其中 } 1 \leqslant k \leqslant N \tag{8-4-6}$$

其逆变换定义为

$$x(k)=\frac{1}{N}\sum_{i=1}^{N} X(i) \mathrm{e}^{2\pi \mathrm{j}(k-1)(i-1)/N}, \text{ 其中 } 1 \leqslant k \leqslant N \tag{8-4-7}$$

快速 Fourier 变换 (fast Fourier transform，FFT) 技术是求解离散 Fourier 变换的最实用、也是最通用的方法。MATLAB 中提供了内在函数 `fft()`，该函数的调用格式很简单，为

`f=fft(x)` 可以进行 FFT，而 `x̂=ifft(f)` 可进行反变换

MATLAB 提供的 `fft()` 函数可以高效地求解 FFT 问题。该函数的另一个显著的特点是它可以对任意长度的向量进行变换，而不要求所变换的向量长度满足 2^n 约束，尽管满足这样长度的变换计算速度快些。

【例 8-30】假设给定数学函数 $x(t)=12\sin(2\pi\times t+\pi/4)+5\cos(2\pi\times 4t)$，选择步长为 h，对其进行 FFT 变换，试绘制出得出变换结果的幅值曲线。对得出的结果试用 FFT 反变换的方法，观察是否能通过反变换还原出所需的信号。

【求解】 对采用的采样周期 h，可以产生 L 个时间值 t_i，并求出这些点上的函数值为 x_i，其相应的频率点可以由 $f_0=1/h, 2f_0, 3f_0, \cdots$ 构成，然后可以由语句

```
>> h=0.01; t=0:h:1; x=12*sin(2*pi*t+pi/4)+5*cos(2*pi*4*t); X=fft(x);
   f=t/h; plot(f(1:floor(length(f)/2)),abs(X(1:floor(length(f)/2))))
   set(gca,'XLim',[0,10])
```

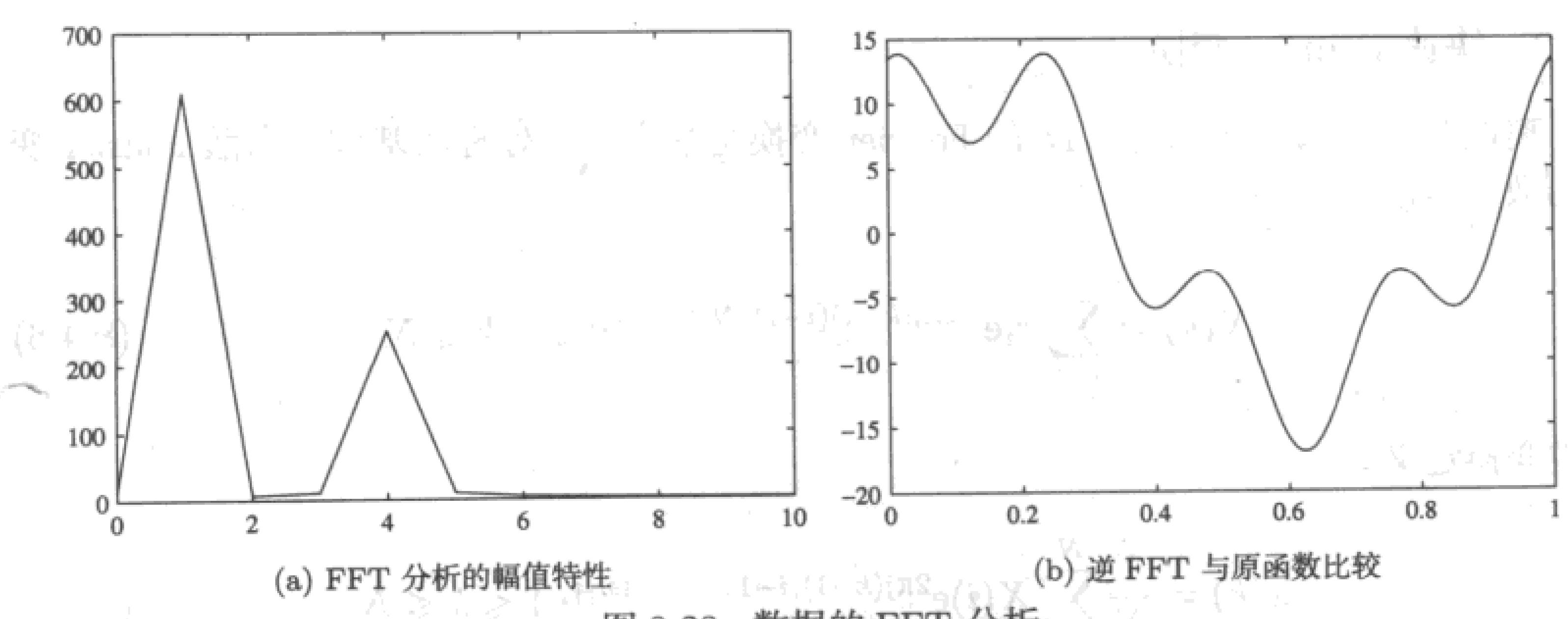
得出 FFT 幅值与频率的关系，如图 8-28 (a) 所示。这里仅取一半数据绘制图形的原因是为了避免 FFT 分析的假频 (aliasing) 现象。分析结果可以看出，在幅值曲线上有两个峰值点，对应的频率值为 10Hz 和 40Hz，正是给定函数中的两个频率值。

快速 Fourier 逆变换可以由 `ifft()` 函数直接求解。

```
>> ix=real(ifft(X)); plot(t,x,t,ix,':'); norm(x-ix)
ans =
     5.626279052291681e-014
```

这样得出的逆 FFT 变换结果与原函数在图 8-28 (b) 中给出。可以看出二者完全一致。由于采用点较稀疏，故曲线看起来不是很光滑。

此外，MATLAB 还提供了二维或更高维的 FFT 与逆 FFT 函数。二维问题可以调用 `fft2()` 和 `ifft2()` 函数，而高维问题可以使用 `fftn()` 和 `ifftn()` 函数。



(a) FFT 分析的幅值特性　　(b) 逆 FFT 与原函数比较

图 8-28 数据的 FFT 分析

8.4.3 滤波技术与滤波器设计

【例 8-31】在介绍滤波技术之前，考虑由曲线 $y(x)=\mathrm{e}^{-x}\sin(5x)$ 叠加标准差为 $\sigma=0.05$ 的零均值的白噪声信号，绘制出噪声污染后的信号曲线。

【求解】下面语句可以得出如图 8-29 所示的曲线。

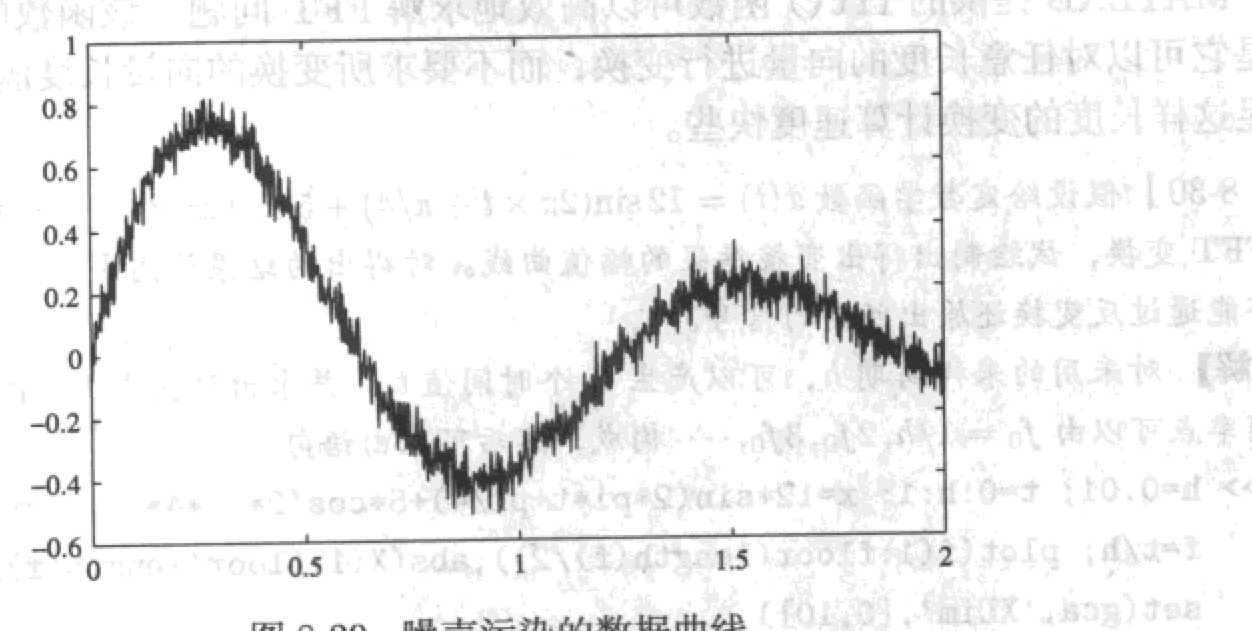


图 8-29 噪声污染的数据曲线

```
>> x=0:0.002:2; y=exp(-x).*sin(5*x); r=0.05*randn(size(x)); y1=y+r; plot(x,y1)
```

对于图 8-29 中给出的被噪声污染的信号曲线，需要引入一种办法消除噪声。例如，可以引入滤波器来实现降低噪声的工作，得出平滑的曲线。

8.4.3.1 线性滤波器的一般模型

线性滤波器模型的一般表达式为

$$H(z)=\frac{b_1+b_2z^{-1}+b_3z^{-2}+\cdots+b_{n+1}z^{-n}}{1+a_2z^{-1}+a_2z^{-2}+\cdots+a_mz^{-m}} \tag{8-4-8}$$

假设输入信号为 $x(n)$，则经过该滤波器后的输出信号可以由下面的差分方程表示为

$$y(k)=-a_1y(k-1)-\cdots-a_my(k-m)+b_1x(k)+b_2x(k-1)+\cdots+b_{n+1}x(k-n) \tag{8-4-9}$$

根据 n 和 m 的不同取值，可以定义出 3 种常用的滤波器，如下：

① FIR **滤波器** 又称为有限长脉冲响应 (finite impulse response，缩写 FIR) 滤波器，需要将式 (8-4-8) 中的 m 值设置成 $m=0$，这时 $\boldsymbol{a}$ 为标量，在控制领域也称移动平均模型 (moving average，简称 MA)，这时用向量 $\boldsymbol{b}$ 就可以表示该滤波器。

② IIR **滤波器** 又称为全极点无限长脉冲响应 (infinite impulse response，IIR) 滤波器，也称为自回归 (auto-regressive，AR) 模型，这时 $n=0$，即 $\boldsymbol{b}$ 为标量，这样用 $\boldsymbol{a}$ 即可以表示该滤波器。FIR 和全极点 IIR 滤波器相比，一般达到同样要求所需的滤波器阶数较高，但其优势是总可以设计出稳定的滤波器[26]。

③ ARMA **滤波器** 又称为一般 IIR 滤波器，也称为自回归移动平均 (auto-regressive moving average，ARMA) 模型，要求 n 与 m 均不为 0，可以用 $\boldsymbol{a},\boldsymbol{b}$ 两个向量表示该滤波器。

假设滤波器可以由 $\boldsymbol{a},\boldsymbol{b}$ 两个向量表示，且假设需要过滤的信号为向量 $\boldsymbol{x}$，则可以调用 `filter()` 函数直接计算出过滤后的信号向量 $\boldsymbol{y}$ 为

```
y=filter(b,a,x)
```

从滤波器的作用又可以分为低通滤波器、高通滤波器和带通滤波器等。顾名思义，低通滤波器是指那些允许低频信号顺利通过，而高频信号被过滤的一类滤波器；而高通滤波器是指高频顺利通过，而低频信号被过滤的滤波器；带通滤波器是指某一个频段的信号被顺利通过，而在这个频段带外的信号均被过滤的滤波器。它们在实际应用中各有其应用。

例如，例 8-31 中给出信号中的噪声为高频的，所以可以考虑设计一个低通滤波器，即频率低时放大倍数接近于 1，高频的信号经过接近于 0 的放大倍数后，就基本可以被滤除。如果滤波器已知，则用 `freqz()` 函数可以对滤波器进行放大倍数分析，即

```
[h,w]=freqz(b,a,N)
```

其中，N 为分析的点数，返回的 $\boldsymbol{h}$ 为复数放大倍数，$\boldsymbol{w}$ 为频率向量。复数放大倍数包含幅值与幅角等信息，若只想获得放大倍数的幅值，则可以用 `plot(w,abs(h))` 命令。

【例 8-32】 假设已经设计出

$$H(z)=\frac{1.2296\times10^{-6}(1+z^{-1})^7}{(1-0.7265z^{-1})(1-1.488z^{-1}+0.5644z^{-2})(1-1.595z^{-1}+0.6769z^{-2})(1-1.78z^{-1}+0.8713z^{-2})}$$

试得出该滤波器的放大倍数，并观察滤波后的信号。

【求解】 可以用下面的语句将滤波器的 $\boldsymbol{b},\boldsymbol{a}$ 向量输入到 MATLAB 环境中，其中，用 `conv()` 函数可以求取多项式乘积。下面的语句还可以绘制出滤波器的放大倍数曲线，如图 8-30 (a) 所示，对低频信号的放大倍数接近 1，不作过滤处理，而对高频信号的放大倍数接近 0，可以将噪声信号滤去。

```
>> b=1.2296e-6*conv([1 4 6 4 1],[1 3 3 1]); a=conv([1,-0.7265],...
     conv([1,-1.488,0.5644],conv([1,-1.595,0.6769],[1,-1.78,0.8713])));
   x=0:0.002:2; y=exp(-x).*sin(5*x); r=0.05*randn(size(x)); y1=y+r;
   [h,w]=freqz(b,a,100); plot(w,abs(h))   % 放大倍数绘制
   figure; y2=filter(b,a,y1);  plot(x,y1,x,y2)   % 滤波效果
```

经过滤波器后的滤波效果如图 8-30 (b) 所示。可见，该滤波器可以较好地对给定噪声信号进行过滤处理，但得出的滤波信号较原信号稍有延迟。

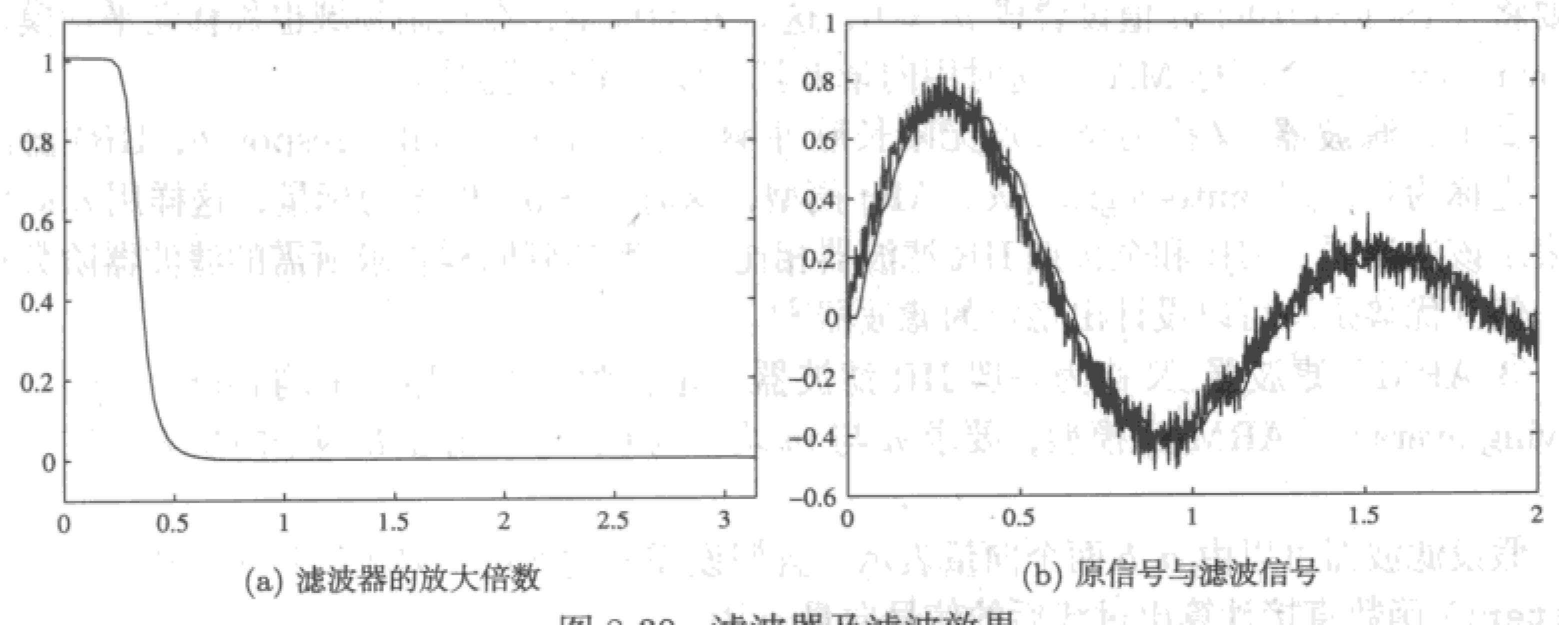


图 8-30　滤波器及滤波效果

8.4.3.2　滤波器设计及 MATLAB 实现

从前面给出的例子可见，如果想较好地对噪声信号进行过滤，则需要用滤波器。滤波器的设计方法多种多样，在 MATLAB 的信号处理工具箱和滤波器设计工具箱中提供了多种滤波器设计的函数，可以直接调用。常用的有两种滤波器类型，如 Butterworth 滤波器和 Chebyshev 滤波器，可以分别用 `butter()` 函数、`cheby1()` 函数 (Chebyshev I 型滤波器) 及 `cheby2()` 函数 (Chebyshev II 型)。它们的调用格式为

```
[b,a]=butter(n,wn),  [b,a]=cheby1(n,r,wn), [b,a]=cheby2(n,r,wn)
```

其中，n 为滤波器的阶次，可以由用户选择，也可以用 MATLAB 的相应函数 (如 `buttord()` 等函数) 设置。w_n 为归一化的频率，定义为实际过滤的频率与信号 Nyquist 频率的比值。假设均匀步距采样的数据个数为 N 个，步距为 Δt则可以计算出基波频率 $f_0 = 1/\Delta t$ Hz，这时 Nyquist 频率的定义为 $f_0N/2$。

【例 8-33】仍考虑例 8-31 中的给定信号，试对阶次及不同的 w_n 值，设计 Butterworth 滤波器并比较滤波效果。

【求解】仍选择 $w_n = 0.1$，用下面的语句则可以设计出不同阶次的 Butterworth 滤波器，并可以绘制出在这些滤波器的放大倍数及滤波效果曲线，如图 8-31 (a)、图 8-31 (b) 所示。从滤波的结果可见，随着 n 的增加，滤波的效果越来越平滑，但延迟也将增大。

```
>> f1=figure; f2=figure;
   for n=5:2:20
      figure(f1); [b,a]=butter(n,0.1); y2=filter(b,a,y1); plot(x,y2); hold on
      figure(f2); [h,w]=freqz(b,a,100); plot(w,abs(h)); hold on
   end
```

选择阶次为 7，对不同的 w_n 可以设计出 Butterworth 滤波器，并绘制出放大倍数及滤波效果曲线，如图 8-32 (a)、图 8-32 (b) 所示。从得出的曲线看，当 w_n 增加时，延迟变小，但得出的滤波效果会明显变差，w_n 太大时滤波没有什么效果。

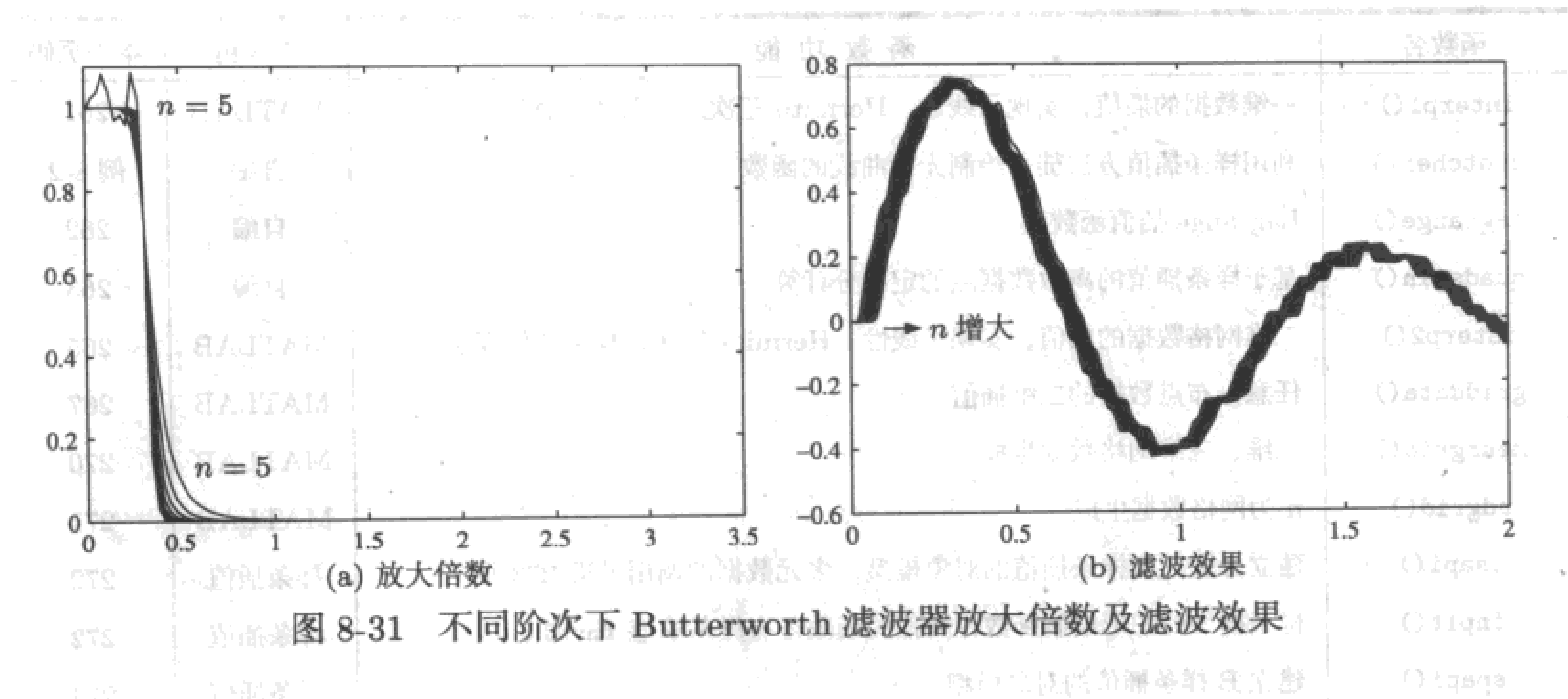


图 8-31 不同阶次下 Butterworth 滤波器放大倍数及滤波效果

```
>> for wn=0.1:0.1:0.7
     figure(f1); [b,a]=butter(7,wn); y2=filter(b,a,y1); plot(x,y2); hold on
     figure(f2); [h,w]=freqz(b,a,100); plot(w,abs(h)); hold on
   end
```

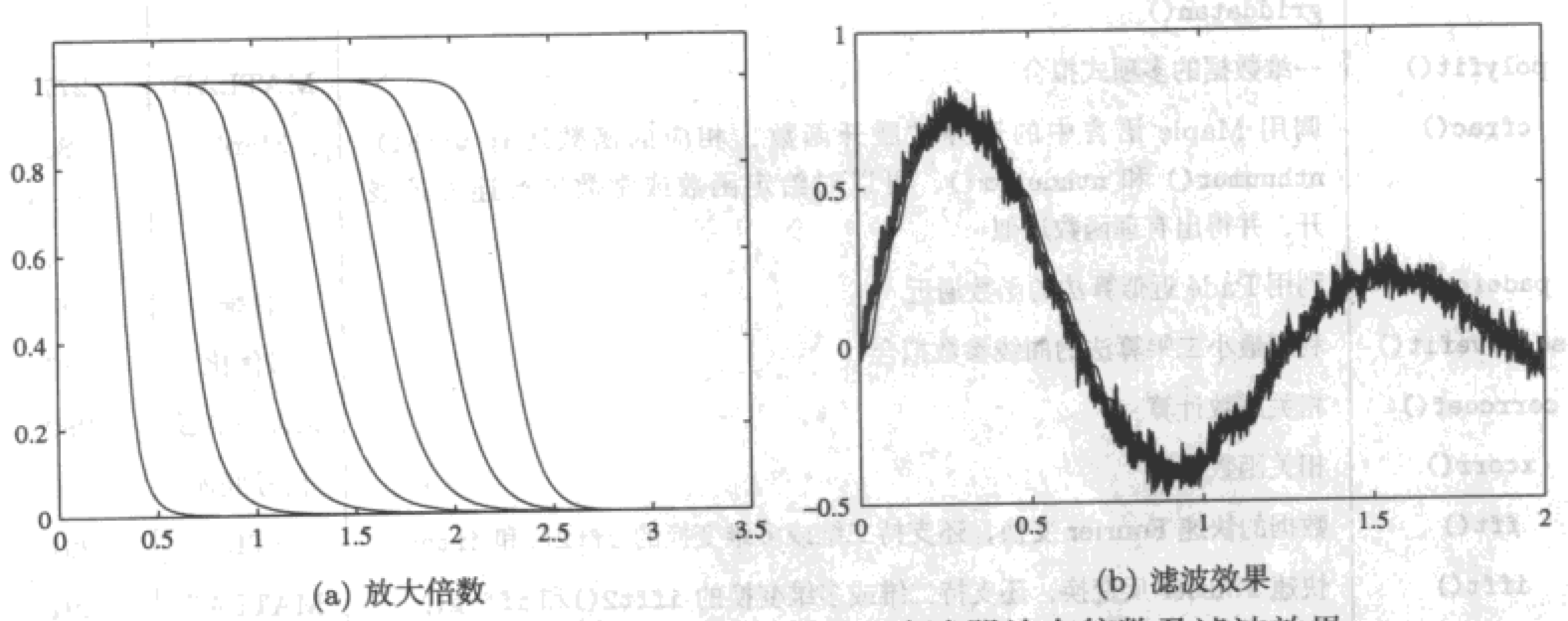


图 8-32 不同滤波频率下 Butterworth 滤波器放大倍数及滤波效果

若想设计高通滤波器，最简单的方法是可以利用 $1-H(z^{-1})$ 直接设计出来，其中，$H(z^{-1})$ 为由前面介绍的方法设计出的低通滤波器。另外，butter() 等函数也可以直接设计高通滤波器和带通滤波器。具体调用格式为

> 高通：$[\boldsymbol{b},\boldsymbol{a}]$=butter($n$,$w_n$,'high')
>
> 带通：$[\boldsymbol{b},\boldsymbol{a}]$=butter($n$,$[w_1,w_2]$)

8.5 本章要点简介

- 本章介绍的数据插值与处理的函数由下表给出。

函数名	函 数 功 能	工具箱	本书页码
interp1()	一维数据的插值，实现了线性、Hermite 三次及样条插值算法	MATLAB	261
sketcher()	利用样条插值方法徒手绘制光滑曲线的函数	自编	例 8-2
lagrange()	Lagrange 插值函数	自编	262
quadspln()	基于样条插值的离散数据点的定积分计算	自编	263
interp2()	二维网格数据的插值，实现了线性、Hermite 三次及样条插值算法	MATLAB	265
griddata()	任意分布点数据的二维插值	MATLAB	267
meshgrid()	二维、三维网格数据生成	MATLAB	270
ndgrid()	n 为网格数据生成	MATLAB	271
csapi()	建立分段三次样条插值的对象模型，多元数据的调用见第 273 页	样条插值	272
fnplt()	样条模型的图形绘制函数，类似的函数还有样条求值 fnval()	样条插值	272
spapi()	建立 B 样条插值的对象模型	样条插值	274
fnder()	基于样条模型的数值微分问题求解函数	样条插值	275
fnint()	基于样条模型的数值积分问题求解函数	样条插值	276
interp3()	三元网格数据的插值处理，还可以用于 n 维数据函数 interpn()	MATLAB	278
griddata3()	三元一般分布数据的插值处理，还提供了可以用于 n 维一般数据插值 griddatan()	MATLAB	278
polyfit()	一维数据的多项式拟合	MATLAB	277
cfrac()	调用 Maple 语言中的连分式展开函数，相应的函数还有 with()，nthnumur() 和 nthdenom()，可以对给定函数或常数进行连分式展开，并得出有理函数近似	Maple	280
padefcn()	利用 Padé 近似算法的函数逼近	自编	283
lsqcurvefit()	利用最小二乘算法的曲线参数拟合	最优化	287
corrcoef()	相关系数计算	MATLAB	289
xcorr()	相关函数计算	信号处理	290
fft()	数据的快速 Fourier 变换，还支持二维或多维变换的 fft2()和 fftn()	MATLAB	291
ifft()	快速 Fourier 反变换，还支持二维或多维变换的 ifft2()和 ifftn()	MATLAB	291
filter()	信号的滤波处理函数	信号处理	293
freqz()	滤波器频域响应分析	信号处理	293
butter()	Butterworth 滤波器设计函数，类似地，还有其他滤波器设计函数，如 I, II 型 Chebyshev 滤波器设计等，函数分别为 cheby1() 和 cheby2()，还可以自动选择滤波器阶次，如使用 buttord() 函数	信号处理	294

- 由已知样本点去计算其他点函数值的方法称为数据插值。本章介绍了一维数据插值的方法及 MATLAB 求解，介绍了曲线平滑处理与基于样本数据的定积分计算，还介绍了二维网格数据及一般分布数据的插值问题求解。
- 着重介绍了两种常用的样条插值方法，如分段三阶样条插值及 B 样条插值方法及应用，并介绍了基于样条模型的微积分运算。

- 介绍了由已知样本点数据获得函数模型的方法，如给定数据的多项式拟合、函数的连分式展开及有理近似、Padé 有理函数逼近、最小二乘曲线拟合方法等。
- 介绍了信号处理的基本内容，如信号的相关系数与相关函数计算、离散信号的快速 Fourier 变换及反变换等，还介绍了线性滤波器的基本概念及基于 MATLAB 语言的 Butterworth 滤波器设计与时域、频域分析。

8.6 习 题

1 用 $y(t)=t^2\mathrm{e}^{-5t}\sin t$ 生成一组较稀疏的数据，并用一维数据插值的方法对给出的数据进行曲线拟合，并将结果与理论曲线相比较。

2 用 $y(t)=\sin(10t^2+3)$ 在 $(0,3)$ 区间内生成一组较稀疏的数据，并用一维数据插值的方法对给出的数据进行曲线拟合，并将结果与理论曲线相比较。

3 用 $f(x,y)=\dfrac{1}{3x^3+y}\mathrm{e}^{-x^2-y^4}\sin(xy^2+x^2y)$ 原型函数生成一组网格数据或随机数据，分别拟合出曲面，并和原曲面进行比较。

4 假设已知一组数据，试用插值方法绘制出 $x\in(-2,4.9)$ 区间内的光滑函数曲线，比较各种插值算法的优劣。

x_i	-2	-1.7	-1.4	-1.1	-0.8	-0.5	-0.2	0.1	0.4	0.7	1	1.3
y_i	0.10289	0.11741	0.13158	0.14483	0.15656	0.16622	0.17332	0.1775	0.17853	0.17635	0.17109	0.16302
x_i	1.6	1.9	2.2	2.5	2.8	3.1	3.4	3.7	4	4.3	4.6	4.9
y_i	0.15255	0.1402	0.12655	0.11219	0.09768	0.08353	0.07015	0.05786	0.04687	0.03729	0.02914	0.02236

5 假设已知实测数据由下表给出，试对 (x,y) 在 $(0.1,0.1)\sim(1.1,1.1)$ 区域内的点进行插值，并用三维曲面的方式绘制出插值结果。

y_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
0.1	0.83041	0.82727	0.82406	0.82098	0.81824	0.8161	0.81481	0.81463	0.81579	0.81853	0.82304
0.2	0.83172	0.83249	0.83584	0.84201	0.85125	0.86376	0.87975	0.89935	0.92263	0.94959	0.9801
0.3	0.83587	0.84345	0.85631	0.87466	0.89867	0.9284	0.96377	1.0045	1.0502	1.1	1.1529
0.4	0.84286	0.86013	0.88537	0.91865	0.95985	1.0086	1.0642	1.1253	1.1904	1.257	1.3222
0.5	0.85268	0.88251	0.92286	0.97346	1.0336	1.1019	1.1764	1.254	1.3308	1.4017	1.4605
0.6	0.86532	0.91049	0.96847	1.0383	1.118	1.2046	1.2937	1.3793	1.4539	1.5086	1.5335
0.7	0.88078	0.94396	1.0217	1.1118	1.2102	1.311	1.4063	1.4859	1.5377	1.5484	1.5052
0.8	0.89904	0.98276	1.082	1.1922	1.3061	1.4138	1.5021	1.5555	1.5573	1.4915	1.346
0.9	0.92006	1.0266	1.1482	1.2768	1.4005	1.5034	1.5661	1.5678	1.4889	1.3156	1.0454
1	0.94381	1.0752	1.2191	1.3624	1.4866	1.5684	1.5821	1.5032	1.315	1.0155	0.62477
1.1	0.97023	1.1279	1.2929	1.4448	1.5564	1.5964	1.5341	1.3473	1.0321	0.61268	0.14763

6 假设已知一组实测数据在文件 c8pdat.dat 中给出，试通过插值的方法绘制出三维曲面。

7 假设已知数据由文件 c8pdat3.dat 给出，其中数据的第 1~3 列分别为 x,y,z 坐标，第 4 列为测出的 $V(x,y,z)$ 函数值，试用三维插值方法对其进行插值。

8 考虑函数 $f(x)=\dfrac{\sqrt{1+x}-\sqrt{x-1}}{\sqrt{2+x}+\sqrt{x-1}}$，试在 $x=3:0.4:8$ 处生成一组样本点，采用分段三次样条和 B 样条分别对数据进行拟合，并由数据结果求取二阶导数，将得出的结果与理论曲线进行比较。

9 已知如下的样本点 (x_i,y_i) 数据，试对其进行分段三次多项式样条插值。

x_i	1	2	3	4	5	6	7	8	9	10
y_i	244.0	221.0	208.0	208.0	211.5	216.0	219.0	221.0	221.5	220.0

10 习题 4 和 5 中给出的数据分别为一元数据和二元数据，试用分段三次样条函数和 B 样条函数对其进行拟合，并求出它们相应函数的数值导数。

11 重新考虑习题 4 中给出的数据，试考虑用多项式插值的方法对其数据进行逼近，并选择一个能较好拟合原数据的多项式阶次。

12 假设习题 4 中给出的数据满足原型 $y(x)=\dfrac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-(x-\mu)^2/2\sigma^2}$，试用最小二乘法求出 μ,σ 的值，并用得出的函数将函数曲线绘制出来，观察拟合效果。

13 试将常数 e 用连分式形式表示，找出能较精确近似其值的项数。

14 试用连分式展开和 Padé 近似方法分别求出下面函数的有理式近似表达式，绘制图形观察拟合效果，并求出具有较好拟合效果的有理式阶次。

① $f(x)=\mathrm{e}^{-2x}\sin 5x$ ② $f(x)=\dfrac{x^3+7x^2+24x+24}{x^4+10x^3+35x^2+50x+24}\mathrm{e}^{-3x}$

15 假设习题 5 中数据的原型函数为 $z(x,y)=a\sin(x^2y)+b\cos(y^2x)+cx^2+dxy+e$，试用最小二乘方法识别出 a,b,c,d,e 的数值。

16 假设已知函数 $f(t)=\mathrm{e}^{-3t}\cos(2t+\pi/3)+\mathrm{e}^{-2t}\cos(t+\pi/4)$，试计算其自相关函数。

17 试求出 Gauss 分布函数 $f(t)=\dfrac{1}{\sqrt{2\pi}3}\mathrm{e}^{-\frac{t^2}{3^2}}$ 的自相关函数，并用 MATLAB 函数生成一组满足 Gauss 分布的伪随机数，用这些数据检验其自相关函数是否和理论值很接近。

18 假设由下面的语句可以生成噪声污染的信号:

```
>> t=0:0.005:5; y=15*exp(-t).*sin(2*t); r=0.3*randn(size(y)); y1=y+r;
```

试求出该信号的 Nyquist 频率，并选择滤波频率，设计出 8 阶 Butterworth 滤波器，使其能有效滤除噪声，又有较小的延迟。

19 高通滤波器可以滤去低频的部分，而保留高频的部分。试为习题 18 中给出的数据设计一个高通滤波器，提取出噪声信号，并和叠加上的实际噪声信号 r 相比较。

第 9 章　概率论与数理统计问题的计算机求解

概率论与数理统计是实验科学中常用的数学分支，其问题的求解是很重要的，但有时也是很繁琐的，传统的方法经常需要用查询表格的方式解决。MATLAB 语言提供了专用的统计学工具箱，其中包含大量的函数，可以直接求解概率论与数理统计领域的问题。第 9.1 节将介绍概率密度、概率分布函数的基本概念及公式，介绍常用概率分布的概率密度函数、概率分布函数的分布曲线绘制，还将介绍基于概率分布的概率运算，并将介绍各种常用分布的伪随机数生成方法，如均匀分布随机数、正态分布随机数、Poisson 分布、Γ 分布、T 分布、F 分布等随机数发生函数。第 9.2 节将介绍一些统计量的计算函数，如数据的均值、方差、矩量、协方差等，并介绍多元随机变量的生成。第 9.3 节将介绍参数估计与区间估计算法及 MATLAB 实现，并介绍多元线性回归问题的求解及非线性函数的最小二乘拟合与求解估计方法。第 9.4 节将介绍假设检验方法，介绍正态分布的均值假设检验、正态性假设检验、给定分布函数的假设检验等。第 9.5 节侧重于方差分析问题及其 MATLAB 求解，介绍单因子方差分析、双因子方差分析等内容及基于 MATLAB 语言的求解方法。

9.1　概率分布与伪随机数生成

9.1.1　概率密度函数与分布函数概述

连续随机变量概率密度一般记作 $p(x)$，概率密度函数满足

$$p(x) \geqslant 0,\ 且\ \int_{-\infty}^{\infty} p(x)\mathrm{d}x = 1 \tag{9-1-1}$$

由概率密度可以定义出概率分布函数

$$F(x) = \int_{-\infty}^{x} p(t)\mathrm{d}t \tag{9-1-2}$$

概率分布函数 $F(x)$ 的物理意义是随机变量 ξ 满足 $\xi \leqslant x$ 发生的概率，该函数为单调递增函数，且满足式子

$$0 \leqslant F(x) \leqslant 1,\ 且\ F(-\infty) = 0,\ F(\infty) = 1 \tag{9-1-3}$$

若已知某概率分布 $f_i = F(x_i)$，需要求出 x_i 的值，在统计学的教材中给出了各种表格，可以根据查表的方法查出所需的 x_i 值。因为分布函数是单调的，所以应该能查询出合适的 x_i 值，这样的问题称为逆分布函数问题。事实上，利用 MATLAB 语言的统计工具箱中提供的函数可以更容易、更精确地求出 x_i 的值。本节将介绍几种常用的概率分布形式，并介绍 MATLAB 的求解函数。

9.1.2 常见分布的概率密度函数与分布函数

9.1.2.1 Poisson 分布

Poisson 分布是一类不同于前述连续分布函数的概率分布，它要求 x 为正整数。Poisson 分布的概率密度为

$$p_p(x) = \frac{\lambda^x}{x!}\mathrm{e}^{-\lambda x},\ x = 0, 1, 2, 3, \cdots \tag{9-1-4}$$

其中，λ 为正整数。

Poisson 分布的概率密度是参数 λ 的函数，且 λ 为正整数。MATLAB 语言的统计工具箱提供了 poisspdf()、poisscdf() 和 poissinv() 函数，可以分别求取 Poisson 分布的概率密度函数、分布函数及逆概率分布的值。这些函数的调用格式为

y=poisspdf(x,λ), F=poisscdf(x,λ), x=poissinv(F,λ)

其中，**x** 为选定的一组横坐标向量，应该由 **x**=0:*k* 语句生成，**y** 为 **x** 各点处的概率密度函数的值。

【例 9-1】 试分别绘制出 $\lambda = 1, 2, 5, 10$ 时 Poisson 分布的概率密度函数与分布函数曲线。

【求解】 仿照前面的例子，可以由下面的语句直接对不同 λ 的值分别调用 poisspdf() 和 poisscdf() 函数，绘制出概率密度函数和分布函数曲线，如图 9-1 (a)、图 9-1 (b) 所示。

```
>> x=[0:15]'; y1=[]; y2=[]; lam1=[1,2,5,10];
   for i=1:length(lam1)
      y1=[y1,poisspdf(x,lam1(i))]; y2=[y2,poisscdf(x,lam1(i))];
   end
   plot(x,y1), figure; plot(x,y2)
```

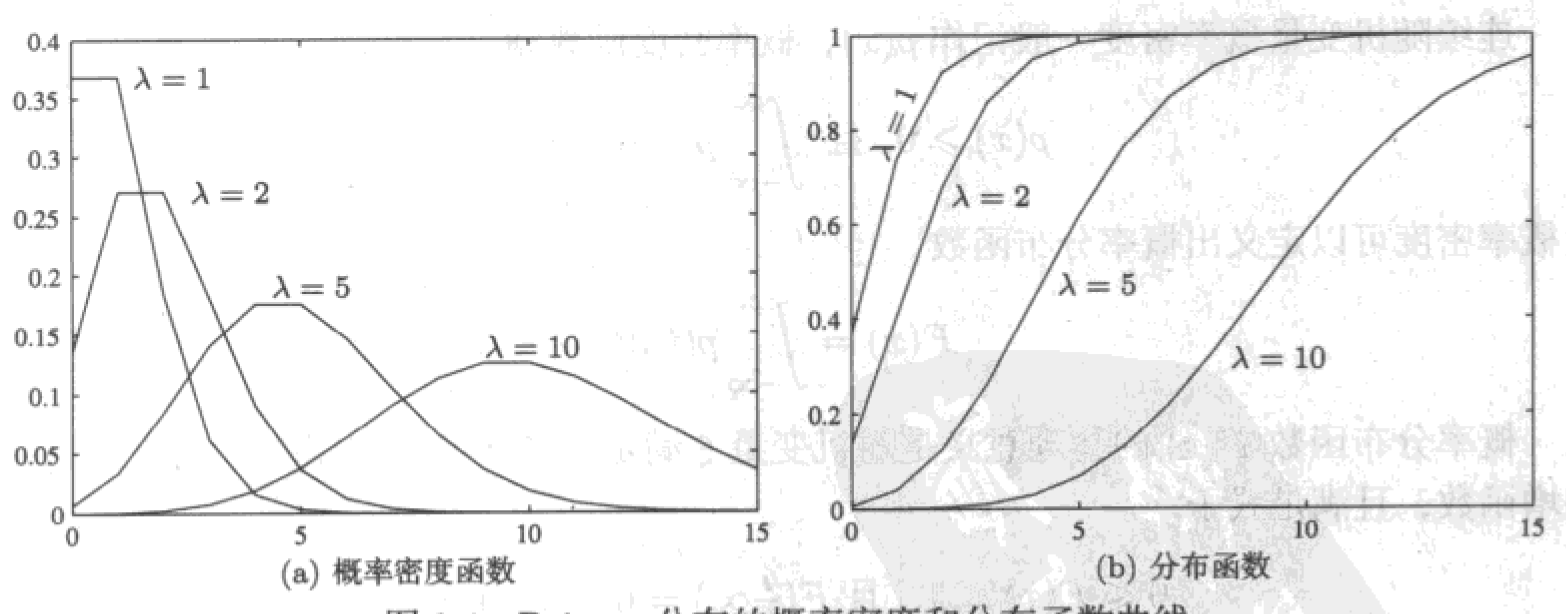


图 9-1 Poisson 分布的概率密度和分布函数曲线

9.1.2.2 正态分布

正态分布的概率密度函数为

$$p_n(x) = \frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{9-1-5}$$

其中，μ 和 σ^2 分别为正态分布的均值和方差，可见概率密度是 μ 和 σ^2 的函数。MATLAB 语言的统计工具箱提供了 normpdf() 和 normcdf()，可以分别求取正态分布的概率密度函数与分布函数的值。另外，若已知概率分布值 $\boldsymbol{F}$，则可以通过 norminv() 函数求出相应的 $\boldsymbol{x}$ 值。这些函数的调用格式为

$\boldsymbol{y}$=normpdf($\boldsymbol{x}$,μ,σ)， $\boldsymbol{F}$=normcdf($\boldsymbol{x}$,μ,σ)， $\boldsymbol{x}$=norminv($\boldsymbol{F}$,μ,σ)

其中，$\boldsymbol{x}$ 为选定的一组横坐标向量，$\boldsymbol{y}$ 为 $\boldsymbol{x}$ 各点处的概率密度函数的值，$\boldsymbol{F}$ 为 $\boldsymbol{x}$ 各点处的分布函数值，故这样得出的三个向量维数应该完全一致。μ 为均值，σ 为标准差，即方差的平方根。MATLAB 的统计学工具箱中提供了大量的此类函数，其后缀名为 pdf 的表示概率密度 (probability density function)，后缀名为 cdf 的为分布函数，又称为累积分布函数 (cumulative distribution function)，后缀为 inv 的表示逆分布函数求解函数。

【例 9-2】 试分别绘制出 (μ,σ^2) 为 $(-1,1),(0,0.1),(0,1),(0,10),(1,1)$ 时正态分布的概率密度函数与分布函数曲线。

【求解】 概率统计类教科书和数学手册中均绘制了某些 μ,σ^2 下的正态分布概率密度曲线和分布函数曲线。学习了 MATLAB 语言及前面介绍的有关函数，读者可以用一个语句精确绘制出任意 μ,σ^2 组合下的正态分布概率密度曲线和分布函数曲线，从而更好地利用实用工具高效解决概率统计问题。

这里给出的问题在 MATLAB 语言中是很容易求解的。首先在 $(-5,5)$ 区间内构造一个横坐标向量 $\boldsymbol{x}$，再定义两个向量，分别表示 μ 和 σ^2 的不同取值，并求出相应的 σ，这样就可以分别调用 normpdf() 和 normcdf() 函数，绘制出概率密度函数和分布函数曲线，如图 9-2 (a)、图 9-2 (b) 所示。

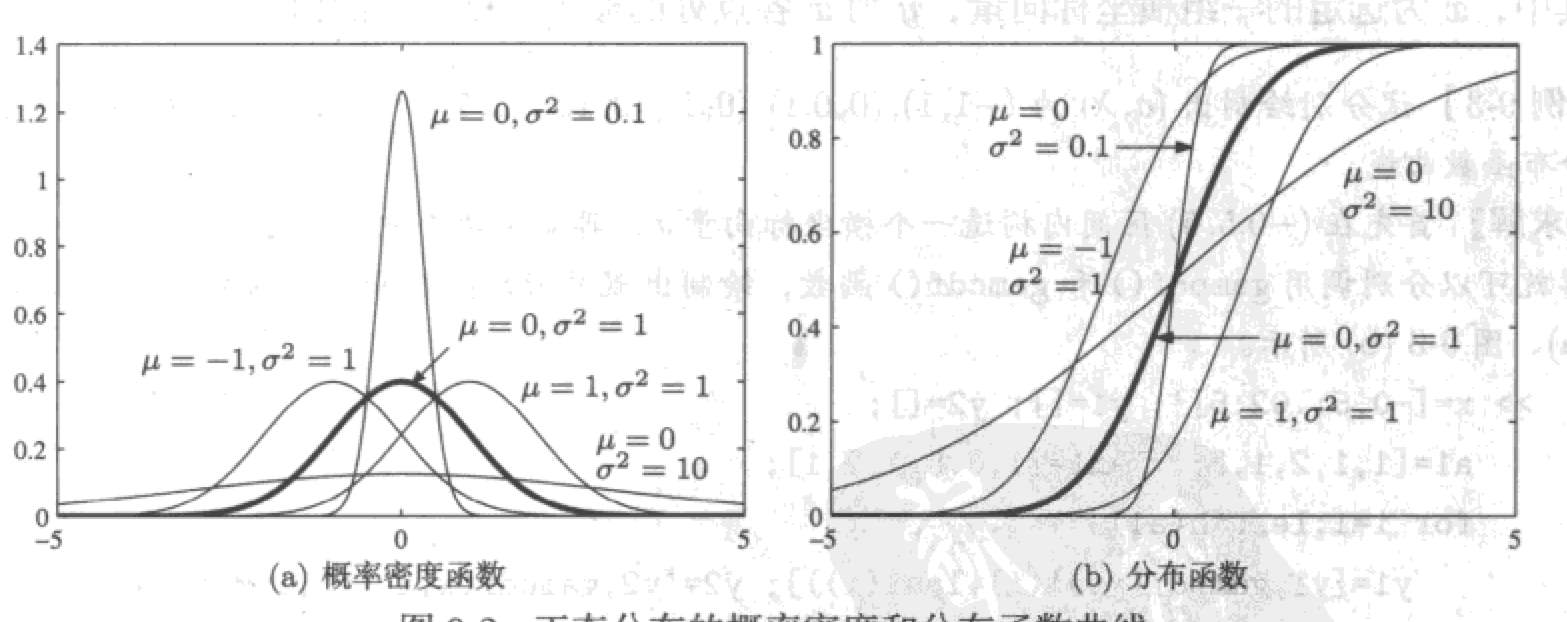


图 9-2 正态分布的概率密度和分布函数曲线

```
>> x=[-5:.02:5]'; y1=[]; y2=[];
   mu1=[-1,0,0,0,1]; sig1=[1,0.1,1,10,1]; sig1=sqrt(sig1);
   for i=1:length(mu1)
      y1=[y1,normpdf(x,mu1(i),sig1(i))]; y2=[y2,normcdf(x,mu1(i),sig1(i))];
   end
   plot(x,y1), figure; plot(x,y2)
```

从得出的曲线可以看出，若 σ^2 相同，则曲线的形状相同，只是对不同的 μ 值进行平移即可。若 σ 不同，则曲线的形状不同，σ^2 的值越小，则概率密度曲线越陡。

$\mu=0,\sigma^2=1$ 的正态分布又称为标准正态分布，其数学表示为 $N(0,1)$。

9.1.2.3 Γ 分布

Γ 分布的概率密度函数为

$$p_\Gamma(x)=\begin{cases}\dfrac{\lambda^a x^{a-1}}{\Gamma(a)}\mathrm{e}^{-\lambda x} & x\geqslant 0\\ 0 & x<0\end{cases} \tag{9-1-6}$$

其中，$\Gamma(a)=\int_0^\infty x^{a-1}\mathrm{e}^{-x}\mathrm{d}x$，$\Gamma(a)$ 函数满足 $\Gamma(a)=a\Gamma(a-1)$，$\Gamma(1)=1$，$\Gamma\left(\dfrac{1}{2}\right)=\pi$，其余的值可以通过积分求得，也可以由 gamma() 函数直接求出。例如，$\Gamma(\pi)$ 的值可以由 gamma(pi) 函数求出为 2.28803779534003。符号运算工具箱也给出了同名函数，可以由 vpa(gamma(sym(pi))) 以任意精度求解该函数值。

Γ 分布的概率密度是参数 a,λ 的函数，MATLAB 语言统计工具箱提供了 gampdf()、gamcdf() 和 gaminv() 函数，可以分别求取 Γ 分布的概率密度函数、分布函数及逆概率分布的值。这些函数的调用格式为

y=gampdf(**x**,a,λ)， **F**=gamcdf(**x**,a,λ)， **x**=gaminv(**F**,a,λ)

其中，**x** 为选定的一组横坐标向量，**y** 为 **x** 各点处的概率密度函数的值。

【例 9-3】 试分别绘制出 (a,λ) 为 $(-1,1),(0,0.1),(0,1),(0,10),(1,1)$ 时 Γ 分布的概率密度函数与分布函数曲线。

【求解】 首先在 $(-0.5,5)$ 区间内构造一个横坐标向量 $\boldsymbol{x}$，再定义两个向量，分别表示 a 和 λ，这样就可以分别调用 gampdf() 和 gamcdf() 函数，绘制出概率密度函数和分布函数曲线，如图 9-3 (a)、图 9-3 (b) 所示。

```
>> x=[-0.5:.02:5]'; y1=[]; y2=[];
   a1=[1,1,2,1,3]; lam1=[1,0.5,1,2,1];
   for i=1:length(a1)
      y1=[y1,gampdf(x,a1(i),lam1(i))]; y2=[y2,gamcdf(x,a1(i),lam1(i))];
   end
   plot(x,y1), figure; plot(x,y2)
```

这里的概率密度图形稍有些问题，因为选择绘图时横坐标步距选择为 0.02，而因为在 −0.02 点处概率密度的值为 0，则在 0 处的函数值为一个非零数值 p_1，所以在图形上看起来在 $x\leqslant 0$ 时函数值不等于 0。事实上，在 $x=0$ 处垂直上升为 p_1，在选择横坐标向量时改用下面的语句即可解决此问题。

```
>> x=[-eps:-0.02:-0.05,0:0.02:5]; x=sort(x');
```

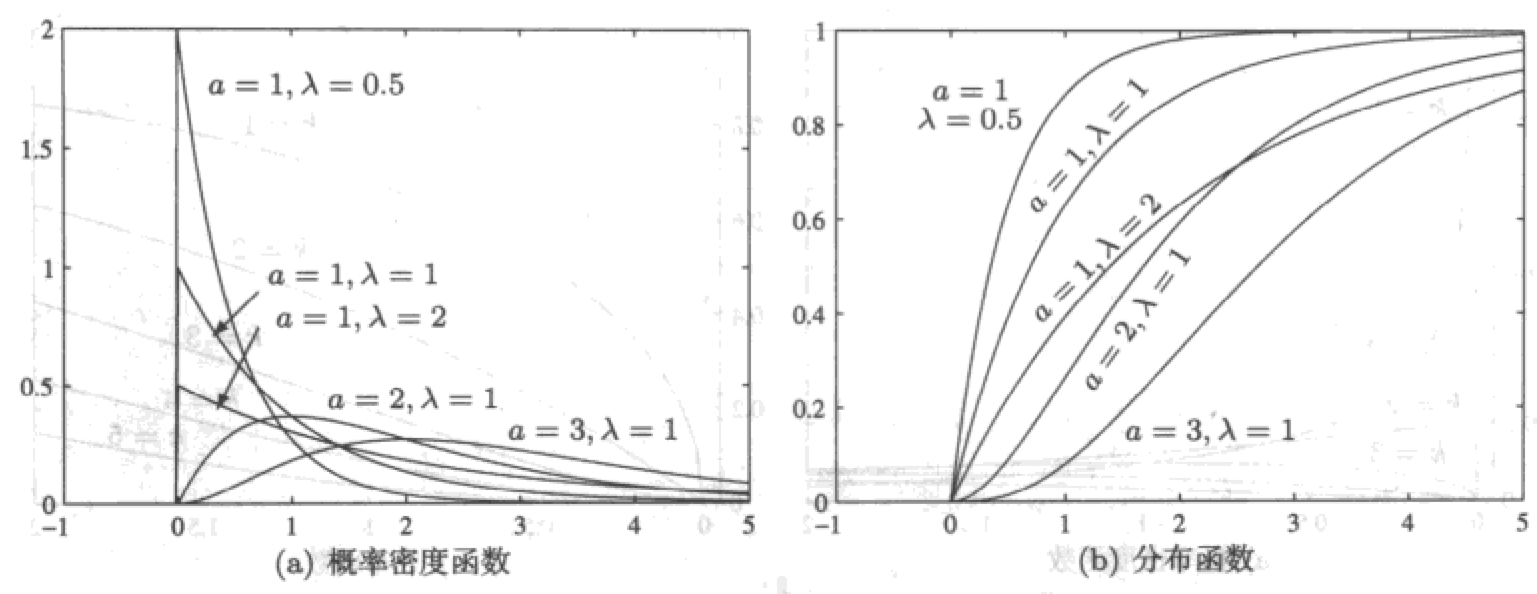


图 9-3 Γ 分布的概率密度和分布函数曲线

9.1.2.4 χ^2 分布

χ^2 分布的概率密度函数为

$$p_{\chi^2}(x)=\begin{cases}\dfrac{1}{2^{\frac{k}{2}}\Gamma\left(\dfrac{k}{2}\right)}x^{\frac{k}{2}-1}\mathrm{e}^{-\frac{x}{2}} & x\geqslant 0\\ 0 & x<0\end{cases} \tag{9-1-7}$$

其中 k 为正整数。可以看出，χ^2 分布是一种特殊的 Γ 分布，其中，$a=k/2$，$\lambda=1/2$。χ^2 分布的概率密度是参数 k 的函数。MATLAB 的统计工具箱提供了 chi2pdf()、chi2cdf() 和 chi2inv() 函数，可以分别求取 χ^2 分布的概率密度函数与分布函数的值。这些函数的调用格式为

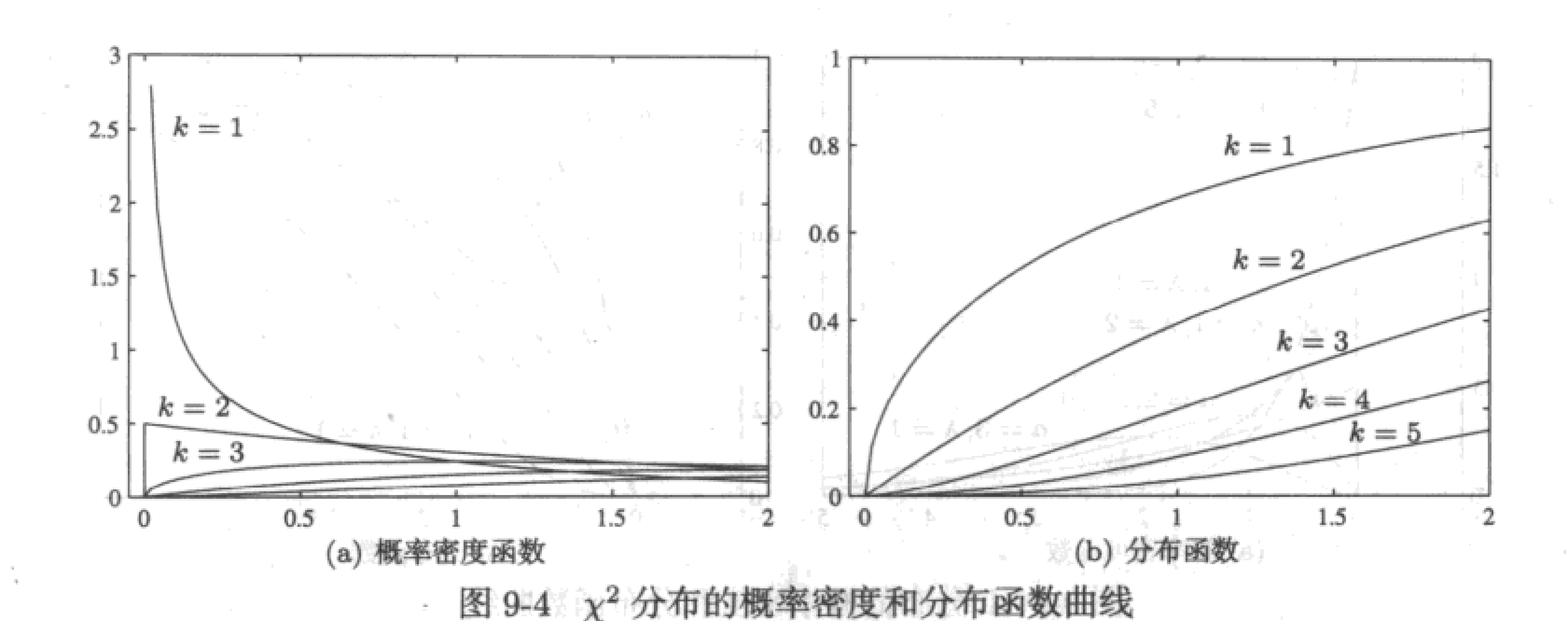
```
y=chi2pdf(x,k), F=chi2cdf(x,k), x=chi2inv(F,k)
```

其中，$\boldsymbol{x}$ 为选定的一组横坐标向量，$\boldsymbol{y}$ 为 $\boldsymbol{x}$ 各点处的概率密度函数的值。

【例 9-4】 试分别绘制出 k 为 $1,2,3,4,5$ 时 χ^2 分布的概率密度函数与分布函数曲线。

【求解】 在 $(-0.05,1)$ 区间内构造一个横坐标向量 $\boldsymbol{x}$，定义一个向量，表示 $\boldsymbol{k}_1$，这样就可以分别调用 chi2pdf() 和 chi2cdf() 函数，绘制出概率密度函数和分布函数曲线，如图 9-4 (a)、图 9-4 (b) 所示。

```
>> x=[-eps:-0.02:-0.05,0:0.02:2]; x=sort(x');
   k1=[1,2,3,4,5]; y1=[]; y2=[];
   for i=1:length(k1)
      y1=[y1,chi2pdf(x,k1(i))]; y2=[y2,chi2cdf(x,k1(i))];
   end
   plot(x,y1), figure; plot(x,y2)
```



(a) 概率密度函数 (b) 分布函数

图 9-4 χ^2 分布的概率密度和分布函数曲线

9.1.2.5 T 分布

T 分布的概率密度函数为

$$p_T(x) = \frac{\Gamma\left(\dfrac{k+1}{2}\right)}{\sqrt{k\pi}\,\Gamma\left(\dfrac{k}{2}\right)}\left(1+\frac{x^2}{k}\right)^{-\frac{k+1}{2}} \tag{9-1-8}$$

T 分布的概率密度是参数 k 的函数，且 k 为正整数。MATLAB 语言的统计工具箱提供了 `tpdf()`、`tcdf()` 和 `tinv()` 函数，可以分别求取 T 分布的概率密度函数、分布函数和逆分布函数的值。这些函数的调用格式为

y=tpdf(x,k)， F=tcdf(x,k)， x=tinv(F,k)

其中，$\boldsymbol{x}$ 为选定的一组横坐标向量，$\boldsymbol{y}$ 为 $\boldsymbol{x}$ 各点处的概率密度函数的值。

【例 9-5】 试分别绘制出 k 为 1,2,5,10 时 T 分布的概率密度函数与分布函数曲线。

【求解】 首先在 $(-5,5)$ 区间内构造一个横坐标向量 $\boldsymbol{x}$，再定义一个 $\boldsymbol{k}_1$ 向量，这样就可以分别调用 `tpdf()` 和 `tcdf()` 函数，绘制出概率密度函数和分布函数曲线，如图 9-5 (a)、图 9-5 (b) 所示。

```
>> x=[-5:0.02:5]'; k1=[1,2,5,10]; y1=[]; y2=[];
   for i=1:length(k1)
      y1=[y1,tpdf(x,k1(i))]; y2=[y2,tcdf(x,k1(i))];
   end
   plot(x,y1), figure; plot(x,y2)
```

9.1.2.6 Rayleigh 分布

Rayleigh 分布的概率密度函数为

$$p_r(x) = \begin{cases} \dfrac{x}{b^2}\mathrm{e}^{-\frac{x^2}{2b^2}} & x \geqslant 0 \\ 0 & x < 0 \end{cases} \tag{9-1-9}$$

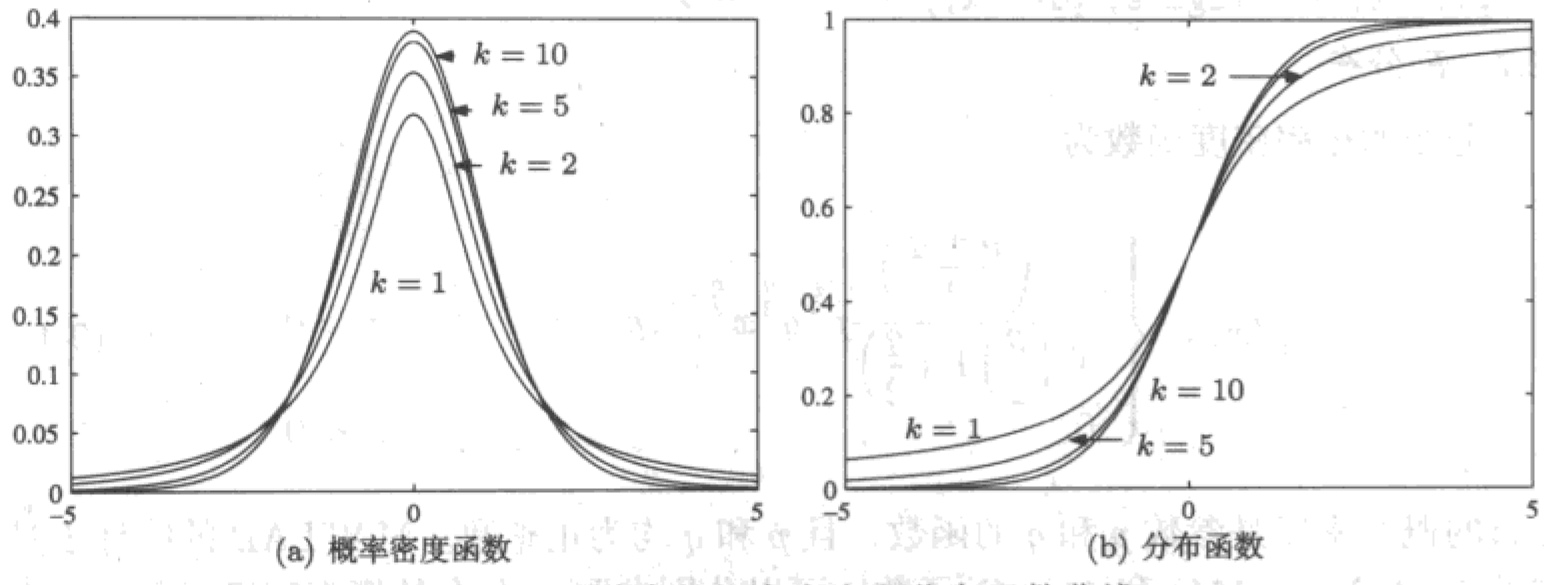


图 9-5 T 分布的概率密度和分布函数曲线

该函数是 b 的函数，MATLAB 语言的统计工具箱提供了 `raylpdf()`、`raylcdf()` 和 `raylinv()`，可以分别求取 F 分布的概率密度函数、分布函数与逆分布函数的值。这些函数的调用格式为

y=raylpdf(x,b), F=raylcdf(x,b), x=raylinv(F,b)

其中，$\boldsymbol{x}$ 为选定的一组横坐标向量，$\boldsymbol{y}$ 为 $\boldsymbol{x}$ 各点处的概率密度函数的值。

【例 9-6】 试分别绘制出 $b=0.5,1,3,5$ 时 Rayleigh 分布的概率密度函数与分布函数曲线。

【求解】 首先在 $(-0.1,5)$ 区间内构造一个横坐标向量 $\boldsymbol{x}$，再定义向量 $\boldsymbol{b}_1$，这样就可以分别调用 `raylpdf()` 和 `raylcdf()` 函数，绘制出概率密度函数和分布函数曲线，如图 9-6 (a)、图 9-6 (b) 所示。

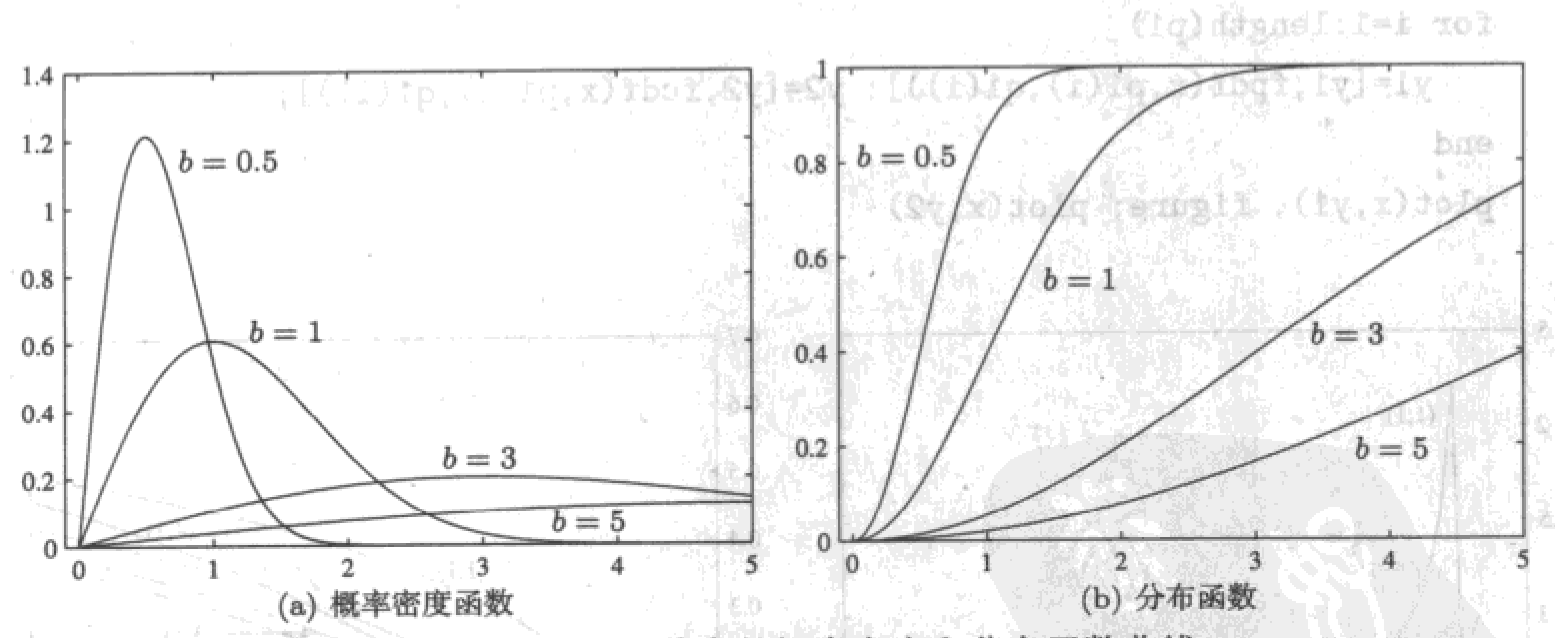


图 9-6 Rayleigh 分布的概率密度和分布函数曲线

```
>> x=[-eps:-0.02:-0.05,0:0.02:5]; x=sort(x');
   b1=[.5,1,3,5]; y1=[]; y2=[];
   for i=1:length(b1)
      y1=[y1,raylpdf(x,b1(i))]; y2=[y2,raylcdf(x,b1(i))];
   end
```

```
plot(x,y1), figure; plot(x,y2)
```

9.1.2.7 *F* 分布

F 分布的概率密度函数为

$$p_F(x)=\begin{cases}\dfrac{\Gamma\left(\dfrac{p+q}{2}\right)}{\Gamma\left(\dfrac{p}{2}\right)\Gamma\left(\dfrac{q}{2}\right)}p^{\frac{p}{2}}q^{\frac{q}{2}}x^{\frac{p}{2}-1}(p+qx)^{-\frac{p+q}{2}} & x\geqslant 0\\ 0 & x<0\end{cases}\tag{9-1-10}$$

F 分布的概率密度是参数 p 和 q 的函数，且 p 和 q 均为正整数。MATLAB 的统计工具箱提供了 fpdf()、fcdf() 和 finv() 函数，可以分别求取 F 分布的概率密度函数、分布函数与逆分布函数的值。这些函数的调用格式为

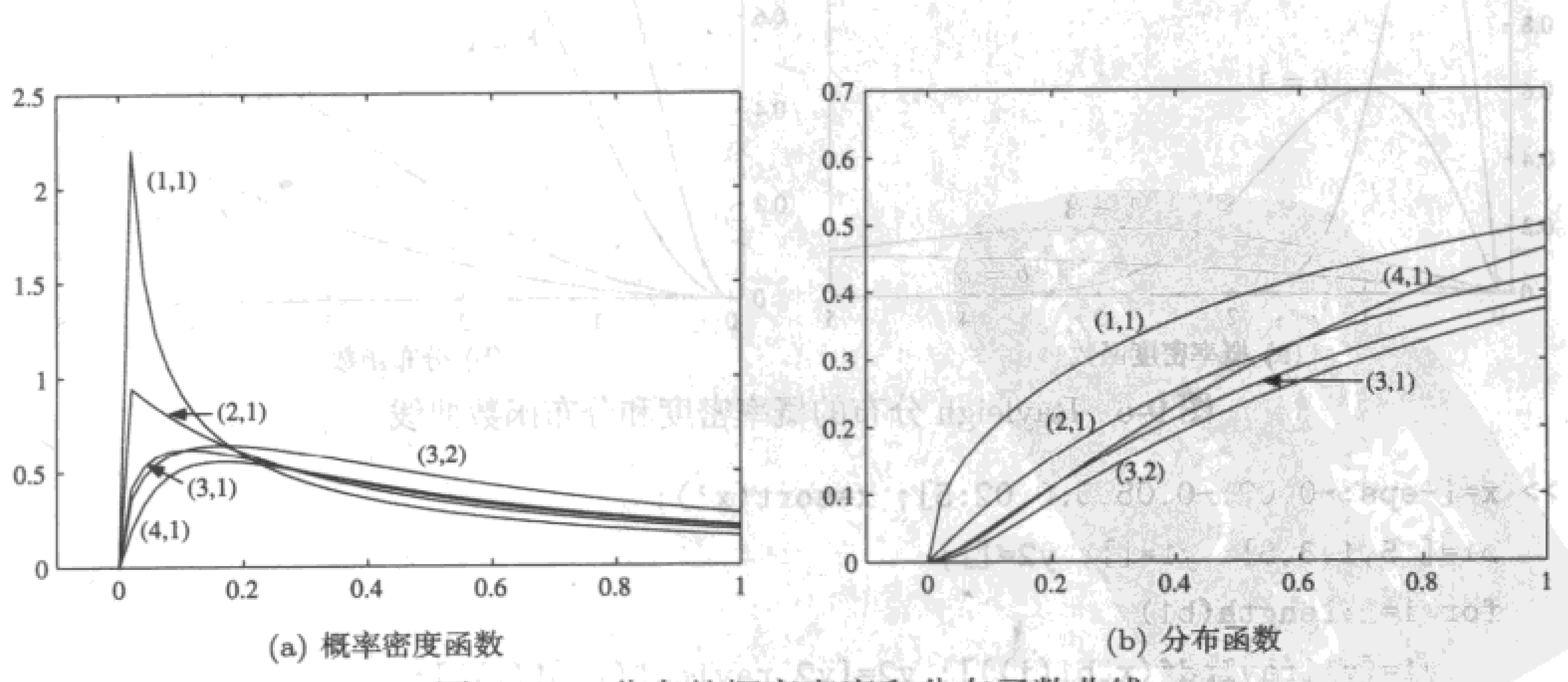
y=fpdf(*x*,*a*,*b*)， *F*=fcdf(*x*,*p*,*q*)， *x*=finv(*F*,*p*,*q*)

其中，$\boldsymbol{x}$ 为选定的一组横坐标向量，$\boldsymbol{y}$ 为 $\boldsymbol{x}$ 各点处的概率密度函数的值。

【例 9-7】 试分别绘制出 (p,q) 对为 $(1,1),(2,1),(3,1),(3,2),(4,1)$ 时 T 分布的概率密度函数与分布函数曲线。

【求解】 首先在 $(-0.1,1)$ 区间内构造一个横坐标向量 $\boldsymbol{x}$，再定义两个向量 $\boldsymbol{p}_1$ 和 $\boldsymbol{q}_1$，这样就可以分别调用 fpdf() 和 fcdf() 函数，绘制出概率密度函数和分布函数曲线，如图 9-7 (a)、图 9-7 (b) 所示。

```
>> x=[-eps:-0.02:-0.05,0:0.02:1]; x=sort(x');
   p1=[1 2 3 3 4]; q1=[1 1 1 2 1]; y1=[]; y2=[];
   for i=1:length(p1)
      y1=[y1,fpdf(x,p1(i),q1(i))]; y2=[y2,fcdf(x,p1(i),q1(i))];
   end
   plot(x,y1), figure; plot(x,y2)
```



(a) 概率密度函数　　(b) 分布函数

图 9-7 *F* 分布的概率密度和分布函数曲线

MATLAB 语言的统计学工具箱还提供了其他各种分布的概率密度函数、分布函数及逆分布函数的函数，可以直接获得各种指定的随机变量分布求取问题。

9.1.3 概率问题的求解

前面介绍过，某随机变量 ξ 分布函数 $F(x)$ 的物理含义是随机变量 ξ 落入 $(-\infty,x)$ 区间的概率，故可以利用分布函数的概念求取满足条件的概率。例如，若想求出 ξ 落入区间 $[x_1,x_2]$ 的概率 $P[x_1\leqslant\xi\leqslant x_2]$，则可以用两个分布函数之差求出。下面给出几个求取概率的公式

$$\begin{cases} P[\xi\leqslant x]=F(x), & \xi\leqslant x\ \text{的概率} \\ P[x_1\leqslant\xi\leqslant x_2]=F(x_2)-F(x_1), & x_1\leqslant\xi\leqslant x_2\ \text{的概率} \\ P[\xi\geqslant x]=1-F(x), & \xi\geqslant x\ \text{的概率} \end{cases} \tag{9-1-11}$$

其中，分布函数可以用前面介绍的分布函数求出。下面将通过例子演示概率求取的方法。

【例 9-8】假设已知某随机变量 x 为 Rayleigh 分布，且 $b=1$，试分别求出该随机变量 x 值落入区间 $[0.2,2]$ 及区间 $[1,\infty)$ 的概率。

【求解】由概率分布函数可以轻易地求出随机变量 x 落入 $(-\infty,0.2]$ 区间和 $(-\infty,2]$ 区间的概率，所以可以由下面的语句立即求出变量落入指定区间的概率为

```
>> b=1; p1=raylcdf(0.2,b); p2=raylcdf(2,b); P1=p2-p1
P1 =
   0.84486339007014
```

另外，由于能直接求出 $(-\infty,1]$ 区间的概率，故可以求出 $x\geqslant 1$ 的概率为

```
>> p1=raylcdf(1,b); P2=1-p1
P2 =
   0.60653065971263
```

【例 9-9】假设二维随机变量 (ξ,η) 的联合概率密度为

$$p(x,y)=\begin{cases} x^2+\dfrac{xy}{3}, & 0\leqslant x\leqslant 1,0\leqslant y\leqslant 2 \\ 0, & \text{其他} \end{cases}$$

试求出 $P(\xi<1/2,\eta<1/2)$。

【求解】已知概率密度 $p(x,y)$，则可以通过积分求出 $P(\xi<x_0,\eta<y_0)$。

```
>> syms x y; f=x^2+x*y/3; P=int(int(f,x,0,1/2),y,0,1/2)
P =
   5/192
```

此处积分下限直接取 0 而不是 $-\infty$ 是因为联合概率密度函数的值在自变量为负的时候为 0，故直接不写出其积分。

9.1.4 随机数与伪随机数

在科学研究和统计分析中经常要用到随机数据。随机数的生成通常有两类方法，其一是依赖一些专用的电子元件发出随机信号，这种方法又称为物理生成法；另一类是通过数学的算法，仿照随机数发生的规律计算出随机数，由于产生的随机数是由数学公式计算出来的，所以这类随机数又称为“伪随机数”。

伪随机数至少有两个优点：首先，若选择相同的随机数种子，这样随机数是可以重复的，这样就创造了重复实验的条件；其次，随机数满足的统计规律可以人为地选择，例如可以自由地选择均匀分布、正态分布、Poisson 分布等，来满足我们的需要。

第 4.1 节中介绍了 `rand()` 和 `randn()` 两个函数，可以分别生成均匀分布伪随机数和正态分布，并介绍了如何生成任意区间的均匀分布伪随机数及任意给定均值、方差的伪随机数生成方法。除了这两类伪随机数之外，在应用中还需要其他各类随机数，如前面介绍的各种概率密度函数对应的随机数。在 MATLAB 环境中，可以用统计工具箱中的语句生成所需的随机数，具体的命令为

A=gamrnd(a,λ,n,m)	生成 $n\times m$ 的 Γ 分布的伪随机数矩阵
B=chi2rnd(k,n,m)	生成 χ^2 分布的伪随机数
C=trnd(k,n,m)	生成 T 分布的伪随机数
D=frnd(p,q,n,m)	生成 F 分布的伪随机数
E=raylrnd(b,n,m)	生成 Rayleigh 分布的伪随机数

【例 9-10】令 $b=1$，试生成 30000×1 个 Rayleigh 分布的随机数，并用直方图检验生成数据的概率分布情况，和理论曲线进行比较。

【求解】由前面的语句可知，由 `raylrnd()` 函数可以立即生成 30000×1 的随机数向量。人为定义一个向量 `xx`，可以用 `hist()` 函数找出随机数落入各个子区间的点个数，并由之拟合出生成数据的概率密度用 `bar()` 函数表示出来。用下面语句可以实现上述功能，并将拟合直方图与理论概率密度在同一坐标系下绘制出来，如图 9-8 所示。可见，生成的数据能较好地满足指定的分布。

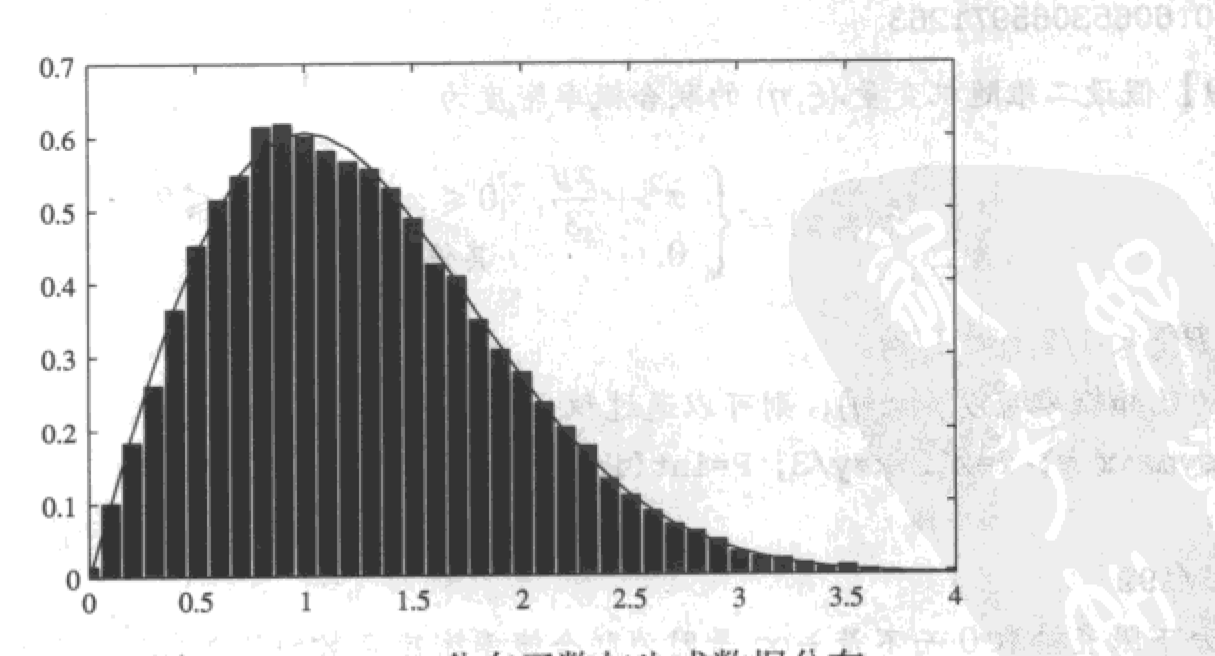


图 9-8 Rayleigh 分布函数与生成数据分布

```
>> b=1; p=raylrnd(1,30000,1);
   xx=0:.1:4; yy=hist(p,xx); yy=yy/(30000*0.1);
   bar(xx,yy), y=raylpdf(xx,1); line(xx,y)
```

9.2 统计量分析

9.2.1 随机变量的均值与方差

假设连续随机变量 x 的概率密度函数为 $p(x)$，则可以如下定义出该变量的数学期望 $\mathrm{E}[x]$ 和方差 $\mathrm{D}[x]$ 为

$$\mathrm{E}[x]=\int_{-\infty}^{\infty}xp(x)\mathrm{d}x,\ \ \mathrm{D}[x]=\int_{-\infty}^{\infty}(x-\mathrm{E}[x])^2p(x)\mathrm{d}x \tag{9-2-1}$$

利用 MATLAB 符号运算工具箱中的积分函数可以求出这两个重要的统计量。

【例 9-11】试用积分方法求取 Γ 分布 $(a>0,\lambda>0)$ 的均值与方差。

【求解】利用 MATLAB 的符号运算工具箱可以立即得出如下结果：

```
>> syms x; syms a lam positive
   p=lam^a*x^(a-1)/gamma(a)*exp(-lam*x); m=int(x*p,x,0,inf)
m =
   1/lam*a
>> s=simple(int((x-1/lam*a)^2*p,x,0,inf))
s =
   a/lam^2
```

亦即，Γ 分布的均值为 $\dfrac{a}{\lambda}$，方差为 $\dfrac{a}{\lambda^2}$。

假设在实际中测出一组样本数据 $x_1,x_2,x_3,\cdots,x_n$，则该随机变量的均值和方差分别定义为

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i,\ \ \hat{s}_x^2=\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2 \tag{9-2-2}$$

由统计学理论可以证明，这样定义的 $\hat{s}_x^2$ 方差是有偏的，所以应该使用无偏的方差，即

$$s_x^2=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2 \tag{9-2-3}$$

并称 $s_x\geqslant 0$ 为标准差。

若已知一组随机变量样本数据构成的向量 $\boldsymbol{x}=[x_1,x_2,x_3,\cdots,x_n]^{\mathrm{T}}$，则可以直接使用 MATLAB 函数 `mean()`、`var()` 和 `std()` 求出该向量各个元素的均值、方差和标准差。这 3 个函数的调用格式为

```
m=mean(x), s2=var(x), s=std(x)
```

这 3 个函数和其他函数还可以处理 $\boldsymbol{x}$ 为矩阵的形式。具体的解释是，对矩阵 $\boldsymbol{x}$ 的每个列

向量进行均值、方差和标准差分析就可以得出一个行向量。若想将矩阵或多维数组 $\boldsymbol{x}$ 全部元素进行统计分析，最简单的格式是 m=mean($\boldsymbol{x}$(:))。

【例 9-12】试生成一组 30000 个正态分布随机数，使其均值为 0.5，标准差为 1.5，试分析这样数据实际的均值、方差和标准差。如果减小随机变量个数，会有什么结果？

【求解】可以用下面的语句生成所需的随机数，并求出该变量的均值、方差和标准差为

```
>> p=normrnd(0.5,1.5,30000,1); [mean(p), var(p), std(p)]
ans =
   0.48792268249878   2.27484783652451   1.50825987035541
```

可见，这样得出的数据均值和方差与理论值比较接近。若减小随机数个数，例如选择 300 个随机数，则可以由以下的语句得出新生成随机数的均值与方差。可见，得出的随机数标准差与理论值相差较大，所以在进行较精确的统计分析时不能选择太少的样本点。

```
>> p=normrnd(0.5,1.5,300,1); [mean(p), var(p), std(p)]
ans =
   0.47448749620068   1.91182304623802   1.38268689378254
```

前面介绍过各种常见的分布函数，如正态分布、Γ 分布等，如果给定了分布，则可以用 MATLAB 统计学工具箱中的现成函数，如 normstat() 或 gamstat() 等函数直接求出该分布的均值和方差。注意，这里的函数命名有些类似于 *pdf.m 文件，即在分布类型标识后加后缀 stat，这样就可以构造出一类求取均值和方差的函数。现在以 gamstat() 函数为例介绍其调用格式为

$[\mu,\sigma^2]$=gamstat(a,λ)

返回的变量为相关分布的均值和方差。

【例 9-13】试求出 Rayleigh 分布 ($b=0.45$) 的均值与方差。

【求解】由于需要求解 Rayleigh 分布，所以需要使用的函数名应该为 raylstat()，可以通过下面的语句直接求出该分布的均值与方差。

```
>> [m,s]=raylstat(0.45)
m =
   0.56399136179198
s =
   0.08691374382403
```

9.2.2 随机变量的矩

假设 x 为连续随机变量，且 $p(x)$ 为其概率密度函数，则可以由下面的式子定义出该变量的 r 阶原点矩和中心矩为

$$\nu_r=\int_{-\infty}^{\infty}x^r p(x)\mathrm{d}x,\ \ \mu_r=\int_{-\infty}^{\infty}(x-\mu)^r p(x)\mathrm{d}x \tag{9-2-4}$$

可见，$\nu_1=\mathrm{E}[x]$，$\mu_2=\mathrm{D}[x]$。

【例 9-14】考虑例 9-11 中 Γ 分布的原点矩和中心矩，并由前几项结果总结一般规律。

【求解】先用下面的语句求解原点矩。

```
>> syms x; syms a lam positive; p=lam^a*x^(a-1)/gamma(a)*exp(-lam*x);
   for n=1:5, m=int(x^n*p,x,0,inf), end
```

得出的结果可以由 LaTeX 表示为

$$\nu_{1-5}=\frac{a}{\lambda},\ \frac{a}{\lambda^2}(a+1),\ \frac{a}{\lambda^3}(a+1)(a+2),\ \frac{a}{\lambda^4}(a+1)(a+2)(s+3),\ \frac{a}{\lambda^5}(a+1)(a+2)(a+3)(a+4)$$

可以总结为

$$\nu_k=\frac{1}{\lambda^k}a(a+1)(a+2)\cdots(a+k-1)=\frac{1}{\lambda^k}\prod_{i=0}^{k-1}(a+k)$$

事实上，该结果也可以由下面的语句直接求出:

```
>> syms n; m=simple(int((x)^n*p,x,0,inf))
m =
   lam^(-n)*gamma(n+a)/gamma(a)
```

同样，可以通过下面的语句求出原问题的中心矩为

```
>> for n=1:7, s=simple(int((x-1/lam*a)^n*p,x,0,inf)), end
```

其 LaTeX 显示为

$$\mu_{1-7}=0,\frac{a}{\lambda^2},\frac{2a}{\lambda^3},\frac{3a(a+2)}{\lambda^4},\frac{4a(5a+6)}{\lambda^5},\frac{5a(3a^2+26a+24)}{\lambda^6},\frac{6a(35a^2+154a+120)}{\lambda^7}$$

但好像这样的积分问题没有规律性的化简结果。

若给定的随机数为一些样本点 $x_1,x_2,\cdots,x_n$，则该随机变量的 r 阶原点矩与中心矩的定义分别为

$$A_r=\frac{1}{n}\sum_{i=1}^{n}x_i^r,\quad B_r=\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^r \tag{9-2-5}$$

MATLAB 语言的统计学工具箱提供了 moment() 函数，可以求出向量 $\boldsymbol{x}$ 的中心高阶矩，但没有直接函数可以求出原点矩。其实，可以用下面的语句求出给定随机向量 $\boldsymbol{x}$ 的 r 阶原点矩与中心矩为

```
A_r=sum(x.^r)/length(x),   B_r=moment(x,r)
```

【例 9-15】仍考虑前面的随机数，可以用下面的语句得出随机数的各阶矩为

```
>> A=[]; B=[]; p=normrnd(0.5,1.5,30000,1); n=1:5;
   for r=n,
      A=[A, sum(p.^r)/length(p)]; B=[B,moment(p,r)];
   end
   A,B
A =
   0.508053449020 2.515471575933 3.545664298043 18.891149742886 40.791198267480
B =
   0 2.25735326887171 -0.02604193983447 15.38146224665522 -1.20870212259271
```

由下面的语句还可以求出各阶矩的理论值。可以看出，从生成的数据求出的各阶矩和理论值的拟合程度也是很好的。

```
>> syms x; A1=[]; B1=[]; p=1/(sqrt(2*pi)*1.5)*exp(-(x-0.5)^2/(2*1.5^2));
   for i=1:5
      A1=[A1,vpa(int(x^i*p,x,-inf,inf),12)];
      B1=[B1,vpa(int((x-0.5)^i*p,x,-inf,inf),12)];
   end
>> A1, B1
A1 =
  [.500000000001,2.50000000000,3.49999999999,18.6250000000,40.8125000000]
B1 =
  [            0.,2.25000000000,             0.,15.1875000000,            0.]
```

9.2.3　多变量随机数的协方差分析

假设随机数 (x_1,y_1), (x_2,y_2), (x_3,y_3), $\cdots$, (x_n,y_n) 为二维随机变量对 (x,y) 的样本，则可以分别定义出二维样本的协方差 s_{xy} 与二维样本的相关系数 η 为

$$s_{xy}=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y}),\quad \eta=\frac{s_{xy}}{s_x s_y} \tag{9-2-6}$$

由上述的式子还可以定义出一个矩阵 $\boldsymbol{C}$ 为

$$\boldsymbol{C}=\begin{bmatrix} c_{xx} & c_{xy} \\ c_{yx} & c_{yy} \end{bmatrix} \tag{9-2-7}$$

式中，$c_{xx}=s_x^2$, $c_{xy}=c_{yx}=s_{xy}$，该矩阵称为协方差矩阵 (covariance matrix)。

多个随机变量的协方差矩阵可以由上述定义扩展出来。MATLAB 中提供了一个专门求解多元随机变量协方差均值的函数 cov()。该函数的调用格式为

$\boldsymbol{C}$=cov($\boldsymbol{X}$)

其中，$\boldsymbol{X}$ 的各列均表示不同的随机变量的样本值。若 $\boldsymbol{X}$ 是向量，则得出的是其方差，否则将返回协方差矩阵 $\boldsymbol{C}$。

【例 9-16】试用 MATLAB 语言产生 4 个满足标准正态分布的随机变量，并求出其协方差矩阵。

【求解】用 MATLAB 给出的 randn() 函数可以生成一个标准正态分布随机数的矩阵。该矩阵有 4 列，表示 4 个不同的随机数变量。该矩阵有 30000 行，表示每个随机数变量均取 30000 个样本点。这样，由下面的语句可以立即得出这 4 个随机数变量的协方差矩阵。可见，该矩阵是对称矩阵，趋近于理论上的单位阵。

```
>> p=randn(30000,4); cov(p)
ans =
   1.00637777855730   0.00125886846413   0.00472604435369  -0.00051901064353
   0.00125886846413   1.00395593527777  -0.00094545007899   0.00480202581544
```

```
  0.00472604435369  -0.00094545007899   1.01104348289978  -0.01189535769006
 -0.00051901064353   0.00480202581544  -0.01189535769006   0.99475563718196
```

9.2.4 多变量正态分布的联合概率密度即分布函数

假设有 n 个正态分布的随机变量 $\xi_1,\xi_2,\cdots,\xi_n$，它们的均值分别为 $\mu_1,\mu_2,\cdots,\mu_n$，可以构成一个均值向量 $\boldsymbol{\mu}$，这些变量的协方差矩阵为 $\boldsymbol{\Sigma}^2$，可以按下面的方式构造出随机数向量为 $\boldsymbol{x}=[x_1,x_2,\cdots,x_2]^{\mathrm{T}}$，这样就可以定义出这些随机变量的联合概率密度为

$$p(x_1,x_2,\cdots,x_n)=\frac{1}{\sqrt{2\pi}}\boldsymbol{\Sigma}^{-1}\mathrm{e}^{-\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-2}\boldsymbol{x}} \tag{9-2-8}$$

MATLAB 语言的统计学工具箱中提供了 mvnpdf() 函数，利用该函数可以计算出多变量正态分布的联合概率密度值。该函数的调用格式为

$[\boldsymbol{p}$=mvnpdf$(\boldsymbol{X},\boldsymbol{\mu},\boldsymbol{\Sigma}^2)]$

其中，$\boldsymbol{X}$ 为 n 列的矩阵，表示各个随机变量的取值，每一列表示一个随机变量，$\boldsymbol{\mu}$ 为每个随机变量均值构成的向量，$\boldsymbol{\Sigma}^2$ 为这些随机变量的协方差矩阵，这样生成的 $\boldsymbol{p}$ 矩阵为列向量，表示每个随机变量组合的概率密度。

【例 9-17】试绘制出均值为 $\boldsymbol{\mu}=[-1,2]^{\mathrm{T}}$，协方差矩阵为 $\boldsymbol{\Sigma}^2=[1,1;1,3]$ 的二维正态分布的联合概率密度函数。若协方差矩阵的非对角线元素为 0，试绘制出新的概率密度函数。

【求解】由于 mvnpdf() 函数只支持一个双列矩阵来表示 $\boldsymbol{X}$，所以应该用适当的转换方法将其转换成两个列向量，再构成两列的矩阵，由该矩阵就可以求出联合概率密度向量，将该向量用 reshape() 函数还原成矩阵形式，最后用三维网格图的形式显示出来，如图 9-9 (a) 所示。

```
>> mu1=[-1,2]; Sigma2=[1 1; 1 3]; % 输入均值向量和协方差矩阵
   [X,Y]=meshgrid(-3:0.1:1,-2:0.1:4); xy=[X(:) Y(:)];  % 产生网格数据并处理
   p=mvnpdf(xy,mu1,Sigma2); P=reshape(p,size(X));      % 求取联合概率密度
   surf(X,Y,P)                     % 绘制联合概率密度的三维网格图
```

对协方差矩阵进行处理，则可以消除协方差矩阵的非对角元素。重新执行下面的语句可以计算出新的联合概率密度函数，如图 9-9 (b) 所示。

```
>> Sigma2=diag(diag(Sigma2)); % 消除协方差矩阵的非对角元素
   p=mvnpdf(xy,mu1,Sigma2); P=reshape(p,size(X)); surf(X,Y,P)
```

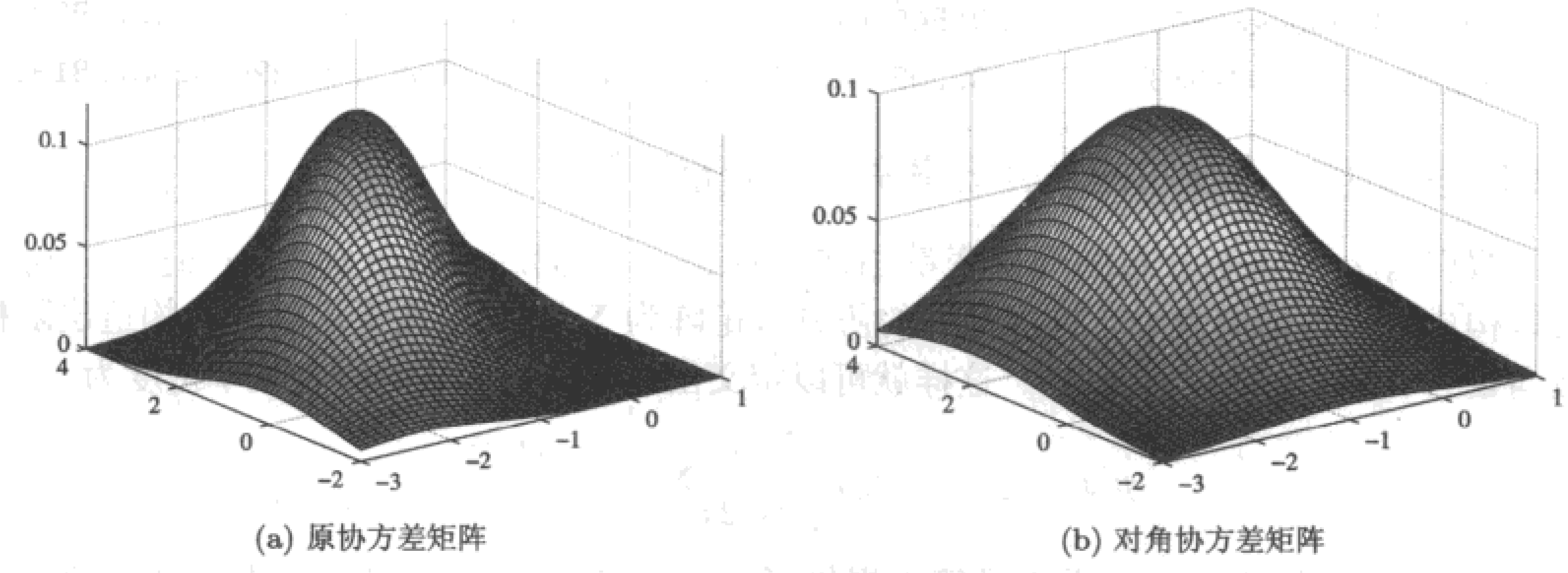
MATLAB 的统计学工具箱还提供了 mvnrnd() 函数，用于产生多变量正态分布随机数。该函数的调用格式为

$\boldsymbol{R}$=mvnrnd$(\boldsymbol{\mu},\boldsymbol{\Sigma}^2,m)$

该函数可以生成 m 组满足多变量正态分布的随机变量，返回的 $\boldsymbol{R}$ 为由 $m\times n$ 矩阵，每一列表示一个随机变量。

【例 9-18】观察例 9-17 中给出的两种二维正态分布的伪随机数分布情况。

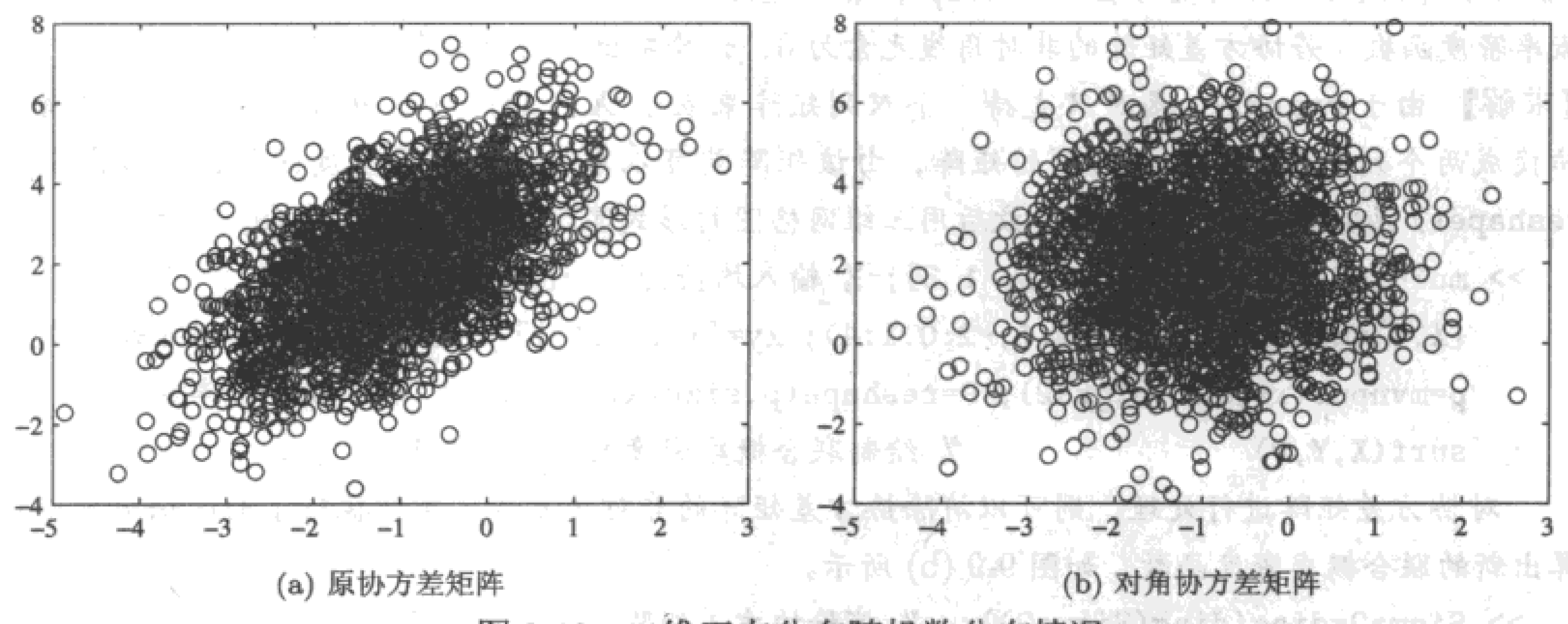
【求解】下面的语句可以得出两种分布下 2000 个点在 x-y 平面上的分布情况，如图 9-10 (a)、图 9-10 (b) 所示。可见，若协方差矩阵为对角矩阵，则两个随机变量之间没有必然联系，所以从分布图看不出随机变量的分布偏向性，而协方差矩阵不是对角矩阵时，随机变量明显有偏向性。



(a) 原协方差矩阵 (b) 对角协方差矩阵

图 9-9 二维正态分布的联合概率密度函数

```
>> mu1=[-1,2]; Sigma2=[1 1; 1 3];
   R1=mvnrnd(mu1,Sigma2,2000); plot(R1(:,1),R1(:,2),'o')
   Sigma2=diag(diag(Sigma2)); figure;
   R2=mvnrnd(mu1,Sigma2,2000); plot(R2(:,1),R2(:,2),'o')
```



(a) 原协方差矩阵 (b) 对角协方差矩阵

图 9-10 二维正态分布随机数分布情况

9.3 数理统计分析方法及计算机实现

9.3.1 参数估计与区间估计

若实测一组数据 $\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$，且已知这些数据满足某种分布，如正态分布，则可以用 MATLAB 语言统计学工具箱中的函数 normfit() 由极大似然法估计出该分布的均值 μ 及方差 σ^2，且同时估计出其置信区间 $\Delta\mu$ 及 $\Delta\sigma^2$。该函数的调用格式为

$[\mu,\sigma^2,\Delta\mu,\Delta\sigma^2]$=normfit($\boldsymbol{x}$,$P_{\mathrm{ci}}$)

其中，P_{ci} 为用户指定的置信度，如 95%，利用该值，此函数会自动调用 norminv() 函数求出相关值，这样就可以得出所需的参数。可以预见，P_{ci} 的值越大 (亦即越趋近于 1)，

则得出的置信区间将越小，亦即得出的结果越接近于真值。

类似于其他概率分布的密度函数等，该工具箱还提供了其他分布的函数，如 Γ 分布的参数估计函数 gamfit()，Rayleigh 分布的参数估计函数 raylfit()，均匀分布的参数估计函数 unifit()，Poisson 分布的参数估计函数 poissfit() 等，这些函数的调用格式很接近，在此不详细介绍。

【例 9-19】 试用 gamrnd() 函数生成一组 $a = 1.5, \lambda = 3$ 的伪随机数，用参数估计的方法以不同的置信度进行估计，比较估计结果。

【求解】 假设生成一组 30000 个数据，并选择置信度为 90%,92%, 95%, 98%，可以用下面的语句得出不同置信度下的参数估计结果。

```
>> p=gamrnd(1.5,3,30000,1); Pv=[0.9,0.92,0.95,0.98]; A=[];
   for i=1:length(Pv)
      [a,b]=gamfit(p,Pv(i)); A=[A; Pv(i),a(1),b(:,1)',a(2),b(:,2)'];
   end
```

为了便于理解，下面用表格的形式将得出的结果显示出来，如表 9-1 所示，表格的全部数据为上述程序中 $\boldsymbol{A}$ 矩阵。可见，置信度选择不同，不会影响参数估计值，但会影响到置信区间的大小。事实上，在一般应用中通常选择 95% 的置信度。

表 9-1　不同置信度下的参数估计结果

置信度	a 参数估计结果			λ 参数估计结果		
	$\hat{a}$	$a_{\min}$	$a_{\max}$	$\hat{\lambda}$	$\lambda_{\min}$	$\lambda_{\max}$
90%	1.506500556	1.505099132	1.507901979	2.991117941	2.987797191	2.994438691
92%	1.506500556	1.505380481	1.507620631	2.991117941	2.988463861	2.993772021
95%	1.506500556	1.505801226	1.507199886	2.991117941	2.98946084	2.992775042
98%	1.506500556	1.506220978	1.506780134	2.991117941	2.990455465	2.991780417

现在考虑随机数数目的不同选择，考虑选择 300,3000,30000,300000,3000000 个随机数，则可以通过下面语句计算出 95% 置信度下参数估计与置信区间的变化。同样，为了更好地显示估计结果，将以列表的形式显示，如表 9-2 所示。

表 9-2　不同随机数个数的参数估计结果

随机数个数	a 参数估计结果			λ 参数估计结果		
	$\hat{a}$	$a_{\min}$	$a_{\max}$	$\hat{\lambda}$	$\lambda_{\min}$	$\lambda_{\max}$
300	1.548677954	1.540991679	1.55636423	2.91172985	2.896265076	2.927194623
3000	1.476057561	1.473908973	1.47820615	3.040589493	3.035438607	3.04574038
30000	1.503327455	1.502624743	1.504030167	2.976242793	2.974591027	2.977894559
300000	1.509546583	1.509323617	1.50976955	2.984774009	2.984252596	2.985295421
3000000	1.498005677	1.497935817	1.498075536	3.006048895	3.005882725	3.006215065

```
>> num=[300,3000,30000,300000,3000000]; A=[];
   for i=1:length(num)
      p=gamrnd(1.5,3,num(i),1);
      [a,b]=gamfit(p,0.95); A=[A;num(i),a(1),b(:,1)',a(2),b(:,2)'];
   end
```

由得出的表格可见，当随机数生成是选择的点较少时，随机数参数的估计效果也不理想，所以在生成随机数时不宜生成太多的点，这在前面均值、方差分析与分布函数分析中已经介绍过了。但也不是随机数点选择得越多越好，因为随机数点选择太多，也不会使参数估计显著提高精度，所以在一般计算中选择 30000 个点即可。

9.3.2 多元线性回归与区间估计

假设输出信号 y 为 n 路输入信号 $x_1, x_2, \cdots, x_n$ 的线性组合为

$$y = a_1x_1 + a_2x_2 + a_3x_3 + \cdots + a_nx_n \tag{9-3-1}$$

其中，$a_1, a_2, \cdots, a_n$ 为待定系数。现在假设已经进行了 m 次实验，实际测出的数据为

$$\begin{array}{cccccc} x_{11} & x_{12} & \cdots & x_{1,n} & \Longrightarrow & y_1 \\ x_{21} & x_{22} & \cdots & x_{2,n} & \Longrightarrow & y_2 \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{m,n} & \Longrightarrow & y_m \end{array} \tag{9-3-2}$$

则可以建立起如下的矩阵方程

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{a} + \boldsymbol{\varepsilon} \tag{9-3-3}$$

式中，$\boldsymbol{a} = [a_1, a_2, \cdots, a_n]^{\mathrm{T}}$ 为待定系数向量，因为每次实验的观测数据可能有误差，故不能完全满足式 (9-3-1)，每一个方程右端均有误差 ε_k，所以 $\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_m]^{\mathrm{T}}$ 为误差构成的向量，$\boldsymbol{y} = [y_1, y_2, \cdots, y_m]^{\mathrm{T}}$ 为各个观测值，且 $\boldsymbol{X}$ 为测出的自变量值，即

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1,n} \\ x_{21} & x_{22} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{m,n} \end{bmatrix} \tag{9-3-4}$$

假设目标函数选择为使得残差的平方和最小，即 $J = \min \boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon}$，则可以得出线性回归模型的待定系数向量 $\boldsymbol{a}$ 的最小二乘估计为

$$\hat{\boldsymbol{a}} = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} \tag{9-3-5}$$

由第 4 章中介绍的线性代数知识可知，MATLAB 语言可以由下面的语句求出最小二乘解，即

a=inv(X'*X)*X'*y　或更简单地，a=X\y

MATLAB 语言的统计学工具箱还提供了多变量线性回归参数估计与置信区间估计的函数 regress()，可以其求出所需的估计结果。该函数的调用格式为

$[\hat{a}, a_{ci}]$=regress(y,X,α)

其中 $1-\alpha$ 为用户指定的置信度。

【例 9-20】假设线性回归方程为 $y=x_1-1.232x_2+2.23x_3+2x_4+4x_5+3.792x_6$，试生成 120 组随机输入值 x_i，计算出输出向量 $\boldsymbol{y}$。以这些信息为已知，观察是否能由最小二乘方法得出待定系数 a_i 的估计值，并得出置信区间。

【求解】本例子用于演示线性回归的方法及 MATLAB 实现，实际应用中应该采用实测数据。由下面的语句可以生成所需的矩阵 $\boldsymbol{X}$ 和向量 $\boldsymbol{y}$，并用最小二乘计算公式得出待定系数向量 $\boldsymbol{a}$ 的估计。

```
>> a=[1 -1.232 2.23 2 4,3.792]'; X=randn(120,6); y=X*a; a1=inv(X'*X)*X'*y
a1 =
    1.00000000000000
   -1.23200000000000
    2.23000000000000
    2.00000000000000
    4.00000000000000
    3.79200000000000
```

可见，因为输出值完全由精确计算得出，所以线性回归参数估计的误差是及其微小的，可以忽略。用 regress() 函数还可以计算出 98% 置信度的置信区间。

```
>> [a,aint]=regress(y,X,0.02)
a =
    1.00000000000000
   -1.23200000000000
    2.23000000000000
    2.00000000000000
    4.00000000000000
    3.79200000000000
aint =
    1.00000000000000    1.00000000000000
   -1.23200000000000   -1.23200000000000
    2.23000000000000    2.23000000000000
    2.00000000000000    2.00000000000000
    4.00000000000000    4.00000000000000
    3.79200000000000    3.79200000000000
```

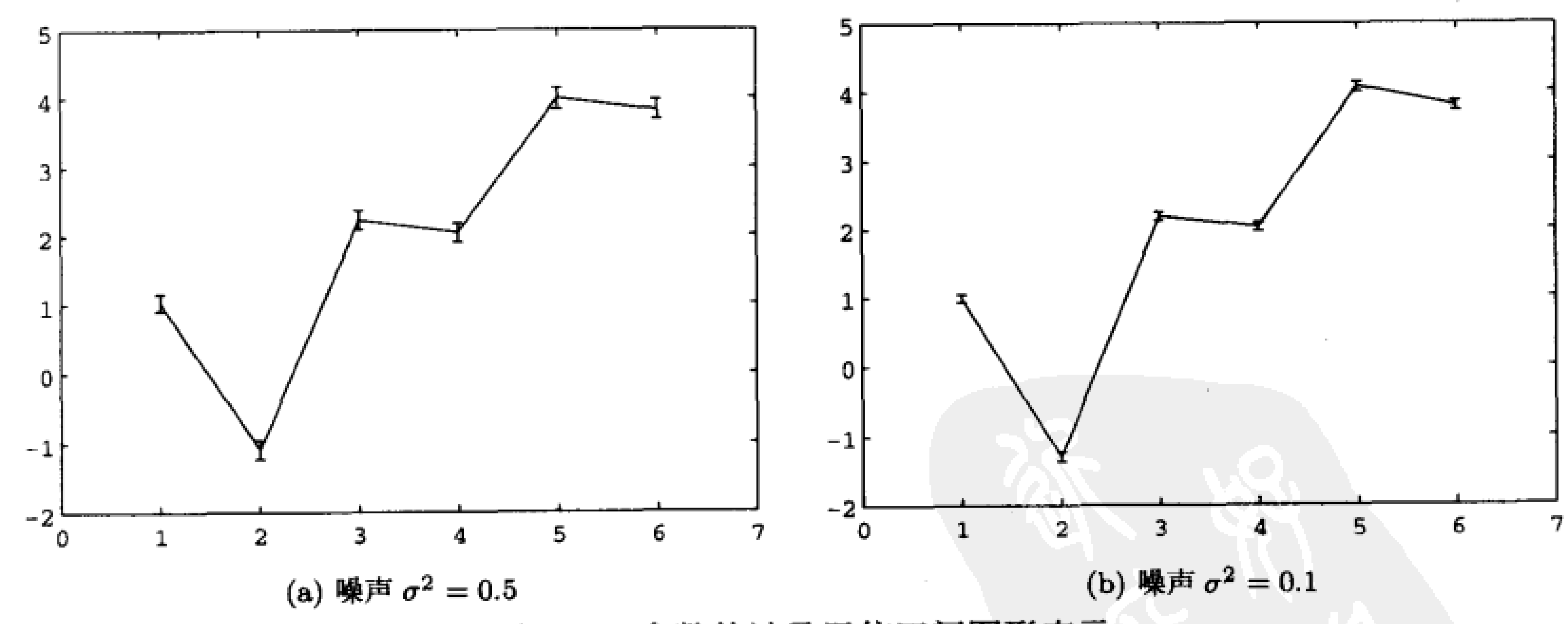
假设观测的输出数据本噪声污染，则可以给输出样本叠加上 $N(0,0.5)$ 区间的正态分布噪声，这时可以用下面语句进行线性回归分析，得出待定系数向量的估计参数及置信区间，用

errorbar() 函数还可以用图形绘制参数估计的置信区间，如图 9-11 (a) 所示。

```
>> yhat=y+sqrt(0.5)*randn(120,1); [a,aint]=regress(yhat,X,0.02)
a =
    1.03887888369425
   -1.20360949260729
    2.19454416841833
    1.89146235051598
   34.06284455872277
    8.70540411609337
aint =
    0.92959508774995    1.14816267963855
   -1.31331402657800   -1.09390495863658
    2.07413724843311    2.31495108840356
    1.77199410730299    2.01093059372896
   33.94183381416145   34.18385530328408
    8.59651534574785    8.81429288643890
>> errorbar(1:6,a,aint(:,1)-a,aint(:,2)-a)
```

所减小噪声的方差，假设方差为 0.1，则可以得出新噪声下参数估计的结果，如图 9-11 (b) 所示。显然估计出的参数更精确。

```
>> yhat=y+sqrt(0.1)*randn(120,1); [a,aint]=regress(yhat,X,0.02);
   errorbar(1:6,a,aint(:,1)-a,aint(:,2)-a)
```



(a) 噪声 $\sigma^2=0.5$　　(b) 噪声 $\sigma^2=0.1$

图 9-11　参数估计及置信区间图形表示

9.3.3　非线性函数的最小二乘参数估计与区间估计

假设有一组数据 $x_i, y_i, i=1,2,\cdots,N$，且已知这组数据满足某一函数原型 $\hat{y}(x)=f(\boldsymbol{a},x)$，其中 $\boldsymbol{a}$ 待定系数向量，但由于误差的影响，可以存在误差，故该函数应该严格

写成 $\hat{y}(x)=f(\boldsymbol{a},x)+\varepsilon$，其中 ε 称为残差，这样可以引入目标函数

$$I=\min_{\boldsymbol{a}}\sum_{i=1}^{N}[y_i-\hat{y}(x_i)]^2=\min_{\boldsymbol{a}}\sum_{i=1}^{N}[y_i-f(\boldsymbol{a},x_i)]^2 \tag{9-3-6}$$

用某种数学算法求出使得目标函数最小的参数 $\boldsymbol{a}$。将估计出的 $\boldsymbol{a}$ 代入原型函数，则可以得出残差 $\varepsilon_i=y_i-f(\boldsymbol{a},x_i)$。类似于第 8.3.5 节中介绍的最小二乘拟合，本节中介绍基于 Gauss-Newton 算法的最小二乘拟合及其 MATLAB 语言 nlinfit()。所不同的是，同时还可以求出残差对估计参数向量 $\boldsymbol{a}$ 的 Jacobi 向量 $\boldsymbol{j}_i$。这些可以用于非线性参数的区间估计函数 nlparci()，得出 95% 置信度下的区间估计结果。这些函数的调用格式为

[$\boldsymbol{a}$,$\boldsymbol{r}$,$\boldsymbol{J}$]=nlinfit($\boldsymbol{x}$,$\boldsymbol{y}$,Fun,$\boldsymbol{a}_0$)	最小二乘拟合
c=nlparci($\boldsymbol{a}$,$\boldsymbol{r}$,$\boldsymbol{J}$)	由置信度为 95% 的置信区间

其中，$\boldsymbol{x}$ 和 $\boldsymbol{y}$ 为实测数据，Fun 为原型函数，可以由 M-函数表示也可以由 inline() 函数表示，$\boldsymbol{a}_0$ 为参数估计的初值。可见，该函数调用中的输入参数与 lsqcurvefit() 函数完全一致。该参数返回的 $\boldsymbol{a}$ 向量为估计出的参数，$\boldsymbol{r}$ 为此参数下的残差构成的向量，$\boldsymbol{J}$ 为各个 Jacobi 行向量构成的矩阵。得出了这些信息，就可以将其用于置信区间的估计，得出置信度为 95% 的置信区间 $\boldsymbol{c}$。下面将通过例子演示非线性函数参数估计与置信区间估计的问题求解。

【例 9-21】试用参数估计的方法重新求解例 8-25 中给出的最小二乘拟合问题，并得出 95% 置信度的置信区间，并在实测信号上叠加均匀分布的噪声信号再进行参数与区间估计。

【求解】假设原型函数为

$$y(x)=a_1\mathrm{e}^{-a_2x}+a_3\mathrm{e}^{-a_4x}\sin(a_5x)$$

其中 a_i 为待定系数。则可以由 inline() 函数直接描述此原型函数，可以表示成 $y=f(\boldsymbol{a},x)$，这样就可以人为指定一组 x_i 值并得出相应的 y_i 值，调用 nlinfit() 函数就可以得出下面的参数估计结果。

```
>> f=inline('a(1)*exp(-a(2)*x)+a(3)*exp(-a(4)*x).*sin(a(5)*x)','a','x');
   x=0:0.1:10; y=f([0.12,0.213,0.54,0.17,1.23],x);
   [a,r,j]=nlinfit(x,y,f,[1;1;1;1;1]); a
a =
   0.11999999763418
   0.21299999458274
   0.54000000196818
   0.17000000068705
   1.22999999996315
```

该函数的拟合结果比 lsqcurvefit() 函数的默认控制结果精确得多，但因为本函数不允许给出精度控制选项，所以也不能得出更精确的结果。用 nlparci() 函数还可以得出 95% 置信度的置信区间，如下所示。可见这样得出的置信区间较小，结果比较精确。

```
>> ci=nlparci(a,r,j)
ci =
   0.11999999712512   0.11999999814323
   0.21299999340801   0.21299999575747
   0.54000000124534   0.54000000269101
   0.17000000036077   0.17000000101332
   1.22999999978603   1.23000000014028
```

现在假设给样本点数据 y_i 叠加上 $[0,0.02]$ 区间均匀分布的随机数，则可以给出如下的语句得出由新样本数据估计出的参数及置信区间。

```
>> y=f([0.12,0.213,0.54,0.17,1.23],x)+0.02*rand(size(x));
   [a,r,j]=nlinfit(x,y,f,[1;1;1;1;1]); a
a =
   0.12281531581639
   0.17072641296744
   0.55113088779121
   0.17347639675132
   1.22916862586480
>> ci=nlparci(a,r,j)
ci =
   0.11857720435195   0.12705342728083
   0.16221631527879   0.17923651065609
   0.54465309442893   0.55760868115349
   0.17055714192171   0.17639565158094
   1.22755955648343   1.23077769524618
>> errorbar(1:5,a,ci(:,1)-a,ci(:,2)-a)
```

这样可以绘制出参数估计及其置信区间，如图 9-12 所示。

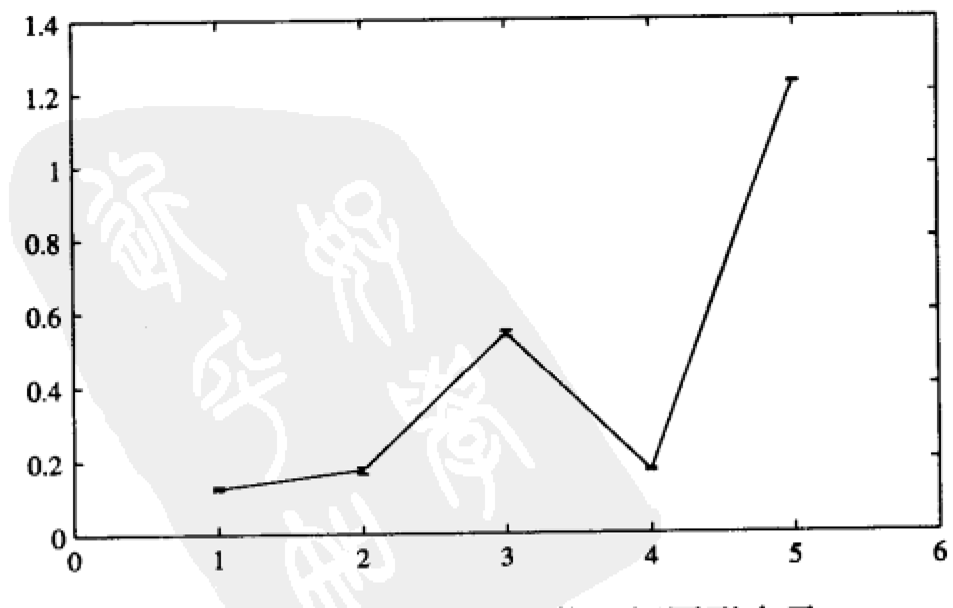


图 9-12 参数估计及置信区间图形表示

【例 9-22】 试利用 nlinfit() 函数求解多变量非线性回归问题。假设非线性函数为

$$f(\boldsymbol{a},\boldsymbol{x}) = (a_1x_1^3 + a_2)\sin(a_3x_2x_3) + (a_4x_2^3 + a_5x_2 + a_6)$$

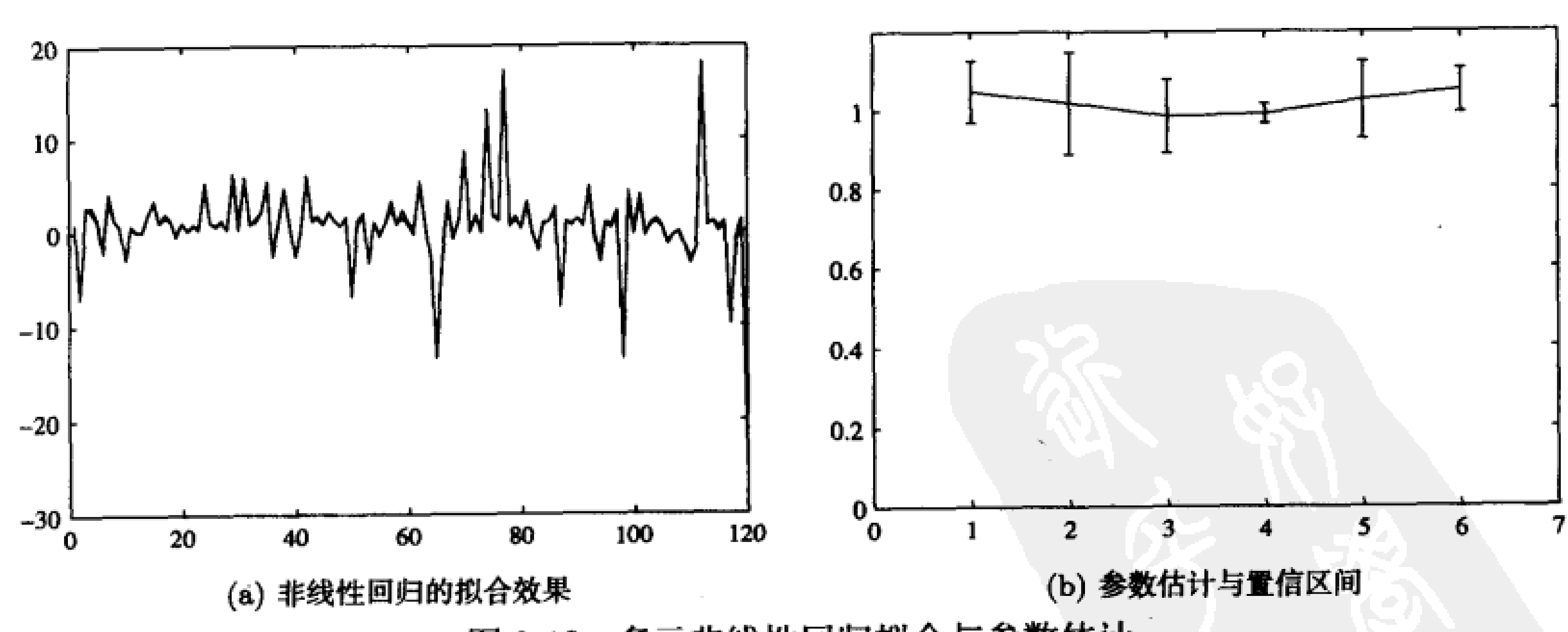
【求解】 假设 a_i 的值均为 1，则可以用下面的语句定义出函数 f，产生一组数据 $\boldsymbol{X}$，则可以计算出一组输出值作为观测数据。

```
>> a=[1;1;1;1;1;1]';
   f=inline(['(a(1)*x(:,1).^3+a(2)).*sin(a(3)*x(:,2).*x(:,3))+',...
     '(a(4)*x(:,3).^3+a(5)*x(:,3)+a(6))'],'a','x');
   X=randn(120,4); y=f(a,X)+sqrt(0.2)*randn(120,1);
```

由这些观测数据可以用非线性回归参数估计函数求出 a_i 的值，并绘制出原观测数据与拟合数据，如图 9-13 (a) 所示，可见拟合结果比较好。从估计的参数看，所得出的结果也是较精确的。

```
>> [ahat,r,j]=nlinfit(X,y,f,[0;2;3;2;1;2]); ahat
ahat =
    1.04839959073146
    1.01882085899938
    0.98446778587739
    0.99107092667601
    1.02519403669663
    1.05040136072101
>> y1=f(ahat,X); plot([y y1])
```

和前面的例子一样，用 nlparci() 函数也能求出来置信区间，也可以用图形的方式表示出 95% 置信度的置信区间，如图 9-13 (b) 所示。



(a) 非线性回归的拟合效果　　(b) 参数估计与置信区间

图 9-13　多元非线性回归拟合与参数估计

```
>> ci=nlparci(ahat,r,j); ci
ci =
    0.96893545453576    1.12786372692717
```

```
    0.88884528207121    1.14879643592754
    0.89167073268603    1.07726483906874
    0.96675355288178    1.01538830047025
    0.92705174408770    1.12333632930555
    0.99419821923406    1.10660450220796
>> errorbar(1:6,ahat,ci(:,1)-ahat,ci(:,2)-ahat)
```

9.4 统计假设检验

9.4.1 统计假设检验的概念及步骤

先假设总体具有某种统计特征 (如具有某种参数或遵从某种分布)，然后再检验这个假设是否可信，这种方法称为统计假设检验方法。统计假设检验在统计学中是有重要地位的。例如，有人提出这样的假设，某灯泡厂生产的某种型号的灯泡平均寿命在 3000 小时以上，如何检验这个假设是否正确。该方法的确切检验方法，即将所有灯泡使用到烧坏为止显然是没有意义的，而应该采用统计假设检验的方法对该假设进行检验。下面将通过例子来演示统计假设检验实现的步骤。

【例 9-23】 已知某产品的平均强度为 $\mu_0 = 9.94$ 公斤，现在改变制作方法，并从新产品中随意抽取 200 件，算得它们的平均强度为 $\bar{x} = 9.73$ 公斤，标准差 $s = 1.62$ 公斤，问制作方法的改变对强度有无显著影响[41]？

【求解】 解决这样的问题需要采用统计假设检验，具体的步骤如下：

① 引入两个命题

$$\begin{cases} \mathscr{H}_0 : \mu = \mu_0 & \text{即无显著改变} \\ \mathscr{H}_1 : \text{拒绝 } \mathscr{H}_0 \text{ 假设} \end{cases}$$

② 选取统计量

$$u = \frac{\sqrt{n}(\bar{x} - \mu_0)}{s} \tag{9-4-1}$$

该统计量满足标准正态分布 $N(0,1)$。对本例来说，可以计算出统计量为

```
>> n=200; mu0=9.94; xbar=9.73; s=1.62; u=sqrt(n)*(mu0-xbar)/s
u =
    1.83323980307623
```

③ 给出显著性水平，由于统计检验毕竟不是确切性检验，所以无论接受还是拒绝该假设都有可能出现错误。引入 α 的意义是判定出现 “取伪” 错误的概率。由于研究的是随机问题，当然不可能令 $\alpha = 0$。一般经常取 $\alpha = 5\%$ 或 $\alpha = 2\%$，用语言表示即为 “可以有 95% (或 98%) 的把握接受或拒绝该假设”。

④ 有了 α 值，则可以用逆正态分布函数求出 $K_{\alpha/2}$ 的值，使得

$$\int_{-K_{\alpha/2}}^{K_{\alpha/2}} \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{x^2}{2}} \mathrm{d}x < 1 - \alpha \tag{9-4-2}$$

该步骤可以用 MATLAB 求解，可以得出不同 α 值下的 $K_{\alpha/2}$ 值，如下所示。其中，第一列为 α 的值，第二列为相应的 $K_{\alpha/2}$ 值。

```
>> alpha=[0.01:0.01:0.05 0.07, 0.09]; K=norminv(1-alpha/2,0,1); [alpha' K']
ans =
    0.01000000000000   2.57582930354890
    0.02000000000000   2.32634787404084
    0.03000000000000   2.17009037758456
    0.04000000000000   2.05374891063182
    0.05000000000000   1.95996398454005
    0.07000000000000   1.81191067295260
    0.09000000000000   1.69539771027214
```

⑤ 由式 (9-4-1) 计算出统计量 u 的值，若 $|u| < K_{\alpha/2}$，则不拒绝 $\mathscr{H}_0$ 假设，否则拒绝该假设。该步骤用 MATLAB 求解为

```
>> abs(u)<K
ans =
      1     1     1     1     1     0     0
```

从比较表达式结果可见，在 $\alpha = 0.01 \sim 0.05$ 的显著性水平下均可以接受假设 $\mathscr{H}_0$，认为改进方法后无显著变化，但增大 α 的值，如这里的 $\alpha = 0.07, 0.09$，则应该拒绝该假设，然而这样结论的可靠性将较低，因为出现错误的概率比较大。

9.4.2 假设检验问题求解

前面的例子介绍了假设检验的 MATLAB 求解。其实，MATLAB 的统计学工具箱中还提供了多个假设检验的函数，例如正态分布均值的假设检验、正态分布性假设检验和任意分布函数的假设检验等。下面介绍这些检验和 MATLAB 统计学工具箱实现。

9.4.2.1 正态分布的均值假设检验

已知某组数据符合正态分布规律，且已知其标准差为 σ。假设其均值为 μ，则可以采用 MATLAB 统计学工具箱的 `ztest()` 函数对该假设进行 Z 假设检验。该函数的调用格式为

$$[H,s,\mu_{\rm ci}]={\tt ztest}(\boldsymbol{X},\mu,\sigma,\alpha)$$

其中，H 为假设检验的结论，当 $H=0$ 时表示不拒绝 $\mathscr{H}_0$ 假设，否则表示拒绝该假设，s 为该检验的显著性水平，$\mu_{\rm ci}$ 为其均值的置信区间。

若未知正态分布的标准差，也可以采用 T 检验法对其进行均值假设检验，调用 `ttest()` 函数对某正态分布的均值进行检验。该函数的调用格式为

$$[H,s,\mu_{\rm ci}]={\tt ttest}(\boldsymbol{X},\mu,\alpha)$$

【例 9-24】 试用正态分布随机数函数生成一组随机数，并对该随机数进行均值假设检验。

【求解】 假设先由 MATLAB 语句生成一组 400 个 $N(1,2^2)$ 的正态分布随机数，由于已知标准差为 2，可以引入假设 $\mathscr{H}_0: \mu=1$，这样可以由下面的 MATLAB 语句进行检验，得出 $H=0$，故可

以接受该假设。

```
>> r=normrnd(1,2,400,1); [H,p,ci]=ztest(r,1,2,0.02)
H =
      0
p =
   0.43594320476001
ci =
   0.84527141065900   1.31054098546717
```

现在试将假设设置为 $\mathscr{H}_0: \mu = 0.5$，则可以给出如下语句，得出 $H = 1$，表示应该拒绝 $\mathscr{H}_0$ 假设。这里得出的置信区间和前面得出的仍然完全一致。

```
>> [H,p,ci]=ztest(r,0.5,2,0.02)
H =
      1
p =
     7.511825163691571e-009
ci =
   0.84527141065900   1.31054098546717
```

若认为标准差未知，则可以采用 T 检验对假设 $\mathscr{H}_0: \mu = 1$ 进行检验，假设检验可以由下面的 MATLAB 语句直接得出，由于得出的 $H = 0$，故表示可以接受该假设。

```
>> [H,p,ci]=ttest(r,1,0.02)
H =
      0
p =
   0.45169871211616
ci =
   0.83635340053445   1.31945899559171
```

9.4.2.2　正态分布假设检验

判定某变量是否为正态分布的传统方法是采用正态概率纸的形式实现的，这时的假设 $\mathscr{H}_0$ 表示待检验的分布是正态分布。其实，这样的过程完全可以用计算机来实现。MATLAB 统计学工具箱中提供了 `jbtest()` 和 `lillietest()` 两个函数，以分别实现 Jarque-Bera 与 Lilliefors 假设检验算法[6]，可以直接由随机样本判定该分布是否为正态分布。这两个函数的调用格式为

$[H,s]$=`jbtest`$(\boldsymbol{X},\alpha)$	Jarque-Bera 检验
$[H,s]$=`lillietest`$(\boldsymbol{X},\alpha)$	Lilliefors 检验

【例 9-25】某工厂生产一种白炽灯，其流明为随机变量 ξ，假设 ξ 满足正态分布 $N(\mu,\sigma^2)$，现从产品中随机抽取 120 个样本，其指标 (流明数) 在表 9-3 给出，试检验正态分布的假设是否正确 (例子及数据来源：文献 [57])。

表 9-3 白炽灯流明实测数据表

216	203	197	208	206	209	206	208	202	203	206	213	218	207	208	202	194	203	213	211
193	213	208	208	204	206	204	206	208	209	213	203	206	207	196	201	208	207	213	208
210	208	211	211	214	220	211	203	216	224	211	209	218	214	219	211	208	221	211	218
218	190	219	211	208	199	214	207	207	214	206	217	214	201	212	213	211	212	216	206
210	216	204	221	208	209	214	214	199	204	211	201	216	211	209	208	209	202	211	207
202	205	206	216	206	213	206	207	200	198	200	202	203	208	216	206	222	213	209	219

【求解】 输入该数据，可以调用现成的 jbtest() 函数或 lillietest() 函数，均能得出 $H=0$，表示可以接受该假设，亦即给出的数据满足正态分布。

```
>> X=[216,203,197,208,206,209,206,208,202,203,206,213,218,207,208,...
      202,194,203,213,211,193,213,208,208,204,206,204,206,208,209,...
      213,203,206,207,196,201,208,207,213,208,210,208,211,211,214,...
      220,211,203,216,224,211,209,218,214,219,211,208,221,211,218,...
      218,190,219,211,208,199,214,207,207,214,206,217,214,201,212,...
      213,211,212,216,206,210,216,204,221,208,209,214,214,199,204,...
      211,201,216,211,209,208,209,202,211,207,202,205,206,216,206,...
      213,206,207,200,198,200,202,203,208,216,206,222,213,209,219];
   [H,p]=jbtest(X,0.05)
H =
     0
p =
   0.72812109026763
```

确定了该数据为正态分布数据，则可以直接用前面介绍的正态分布拟合函数 normfit() 求出该分布的均值、方差及其置信区间。

```
>> [mu1,sig1,mu_ci,sig_ci]=normfit(X,0.05); mu=[mu1,mu_ci']
mu =
   208.8167   207.6737   209.9596
>> sig=[sig1, sig_ci']
sig =
   6.32320434315278   5.61178285217307   7.24282586089008
```

【例 9-26】 试用统计学工具箱生成一组 Γ 分布数据，用现成函数验证其是否为正态分布数据，显然这些数据不是正态分布的，所以假设检验结果应该是 1。

【求解】 给出下面的语句，先用 MATLAB 语句生成一组 Γ 分布的随机数，然后调用 jbtest() 函数立即得出如下结果。

```
>> r=gamrnd(1,3,400,1); [H,p,c,d]=jbtest(r,0.05)
H =
```

```
      1
  p =
      0
```

即 $\mathscr{H}_0$ 应该被拒绝。

9.4.2.3 其他分布的 Kolmogorov-Smirnov 检验

前面介绍的 Jarque-Bera 与 Lilliefors 假设检验算法和 MATLAB 函数只能用于检验某分布是否为正态分布，却不能用于其他分布的检验。Kolmogorov-Smirnov 检验是检验任意已知分布函数的一种有效的假设检验算法。MATLAB 的统计学工具箱中提供了 kstest() 函数，实现了该算法，其调用格式为

```
[H,s]=kstest(X,cdffun,α)
```

其中，cdffun 为两列的均值，第 1 列为自变量，第 2 列应该为要检验的分布函数在自变量处的值。在构造 cdffun 时可以用现成分布函数求取，也可以自己按需要检验的分布函数编写，所以可以用此算法检验是否为任意给定的分布。

【例 9-27】 试对例 9-26 中生成的随机数进行假设检验。该随机数满足 Γ 分布。

【求解】 首先假设其满足 Γ 分布，则由 gamfit() 函数可以得出其两个参数 a,λ。

```
>> r=gamrnd(1,3,400,1); alam=gamfit(r)
alam =
    1.04562614530773    3.28680216836236
```

这样就能构造出 Γ 分布的分布函数为 gamcdf(sort(r),alam(1),alam(2))。将其代入 kstest() 函数，就可以对前面的假设进行检验。

```
>> r=sort(r);
   [H0,p]=kstest(r,[r gamcdf(r,alam(1),alam(2))],0.05)
H0 =
      0
p =
      0.87724898430408
```

由于得出的 $H=0$，所以可以认为通过假设检验，表明前面生成的数据确实是 Γ 分布，且其 $\hat{a}=1.0456, \hat{\lambda}=3.2868$ (真值为 $a=1,\lambda=3$)。

9.5 方差分析及计算机求解

方差分析 (analysis of variance，ANOVA) 是英国统计学家兼遗传学家 Fischer R A 提出的一种分析方法，在医学研究、科学试验和现代工业质量控制等众多领域有着广泛的应用。

试验样本的分组方式不同，则采用方差分析方法也不同，一般常用的分组方法是单向分组 (one-way)、两向分组 (two-way) 和 n 向分组 (N-way) 方法。下面将分别介绍各种形式下的方差分析方法及其 MATLAB 实现。

9.5.1 单因子方差分析

顾名思义，单因子方差分析就是指对一些观察来说，只有一个外界因素可能对观测的现象产生影响。假设需要研究 N 种药物对某病症的疗效，可以采用这样的方法。将病人随机地分成 N 组，每组有 m 个病人，这样将每个病人的疗效观测指标 (如治愈需要的天数) 记作 $y_{i,j}$，其中下标 i 表示第 i 组，$(i=1,2,\cdots,N)$，j 表示某组内病人的编号，$(j=1,2,\cdots,m)$，则第 i 组第 j 个病人的观测指标即为 $y_{i,j}$。现在仿照 MATLAB 的冒号表达式记号，记第 i 组的所有病人观测指标为 $y_{i,:}$，或各组的第 j 个病人观测指标为 $y_{:,j}$，则这样得出的表示均为向量。还可以引入平均值的概念，例如用 $\bar{y}_{i,:}$ 表示第 i 组内病人观测指标的平均值，用 $\bar{y}_{:,:}$ 表示所有组内所有病人观测指标的平均值，则可以构造出如表 9-4 所示的标准方差分析表，根据该表格中给出的数据找出所需的规律。

表 9-4 单因子方差分析表

方差来源	平方和	自由度	均 方	F	p 值
因子效应	$\mathrm{SSA}=\sum_i n_i\bar{y}_{i,:}^2-N\bar{y}_{:,:}^2$	$I-1$	$\mathrm{MSSA}=\mathrm{SSA}/(I-1)$	MSSA/MSSE	$p=P(F_{I-1,N-I}>c)$
随机误差	$\mathrm{SSE}=\sum_i\sum_k y_{i,k}^2-\sum_i n_i\bar{y}_{i,:}^2$	$N-I$	$\mathrm{MSSE}=\mathrm{SSE}/(N-I)$		
和	$\mathrm{SST}=\sum_i\sum_k y_{i,k}^2-N\bar{y}_{:,:}^2$	$N-1$			

上面所采用的药物作为分组的依据，称为因子 (factor)，它们的差异 (如这里采用药物的不同) 称为因子的水平。因为这里始终将药物作为影响观测指标的因素，故称为单因子分析。这里仍然使用假设检验的方法进行方差分析，假设的 $\mathscr{H}_0$ 为各组的平均观测指标是相同的。表格中比较重要的数值是最后两列，Fisher 分布的值 F 和置信度为 c 时的概率值 p，概率值可以通过逆分布函数求出。如果得出的概率值 $p<\alpha$，α 为置信度，则应该拒绝假设 $\mathscr{H}_0$，否则不拒绝假设。

MATLAB 的统计学工具箱提供了 anova1() 函数，可以用于对给出的数据进行单因子方差分析。该函数的调用格式为

```
[p,tab,stats]=anova1(X)
```

其中，$\boldsymbol{X}$ 为需要分析的数据，该数据应该为一个 $m\times n$ 矩阵，每一列对应于随机分配的一个组的测试数据，这样就会返回概率 p，方差表数据 tab，其内容如表 9-4 所示，stats 为统计结果量，为结构体变量，包括每组的均值等信息。该函数还将自动打开两个 MATLAB 图形窗口，一个按表 9-4 的形式显示出该表格的内容，另一个图形窗口将显示盒式图。

【例 9-28】设有 5 种治疗某病的药物，要比较它们的疗效，假定将 30 个病人随机地分成 5 组，每组 6 人，令每组病人使用同一种药物，并记录病人从使用药物开始到痊愈的时间，如表 9-5 所示，试评价疗效有无显著差异 (例子及数据来源：文献 [23])。

【求解】 根据给出的表格，可以按规则立即建立起 $\boldsymbol{A}$ 矩阵，先求出各列的均值，并对各组数据进行单因子方差分析，得出如下的方差分析结果。

表 9-5　实验数据表 (治愈的天数)

病人编号	药物 1	药物 2	药物 3	药物 4	药物 5	病人编号	药物 1	药物 2	药物 3	药物 4	药物 5
1	5	4	6	7	9	2	8	6	4	4	3
3	7	6	4	6	5	4	7	3	5	6	7
5	10	5	4	3	7	6	8	6	3	5	6

```
>> A=[5,4,6,7,9; 8,6,4,4,3; 7,6,4,6,5; 7,3,5,6,7; 10,5,4,3,7; 8,6,3,5,6];
   mean(A)
ans =
   7.50000000000  5.00000000000  4.33333333333  5.16666666667  6.16666666667
>> [p,tbl,stats]=anova1(A)
p =
   0.01359036533623
tbl =
  'Source'     'SS'        'df'    'MS'       'F'       'Prob>F'
  'Columns'    [36.4667]   [ 4]    [9.1167]   [3.8960]  [0.0136]
  'Error'      [58.5000]   [25]    [2.3400]         []        []
  'Total'      [94.9667]   [29]          []         []        []
stats =
  gnames: [5x1 char]
       n: [6 6 6 6 6]
  source: 'anova1'
   means: [7.500000000000 5 4.333333333333 5.166666666667 6.166666666667]
      df: 25
       s: 1.52970585407784
```

同时，anova1() 函数还将自动打开两个图形窗口，分别绘制出方差分析表和盒式图，如图 9-14 (a)、图 9-14 (b) 所示。由于得出的概率值 $p = 0.0136 < \alpha$，其中 $\alpha = 0.02$ 或 0.05，故应该拒绝给出的假设，认为这些药物确实对治愈时间有显著影响。该结果和文献中由统计软件 SAS 求出的结果完全一致。事实上，从得出的盒式图可以看出，第 3 种药物的治愈时间显然低于第 1 种药物。

9.5.2　双因子方差分析

如果有两种因子可能影响到某现象的统计规律，则应该引入双因子方差分析的概念。这时观测量 y 可以表示为一个三维数组 $y_{i,j,k}$，表示第 1 个因子取第 i 个水平，第 2 个因子取第 j 个水平时，组内第 k 个对象的观测指标。

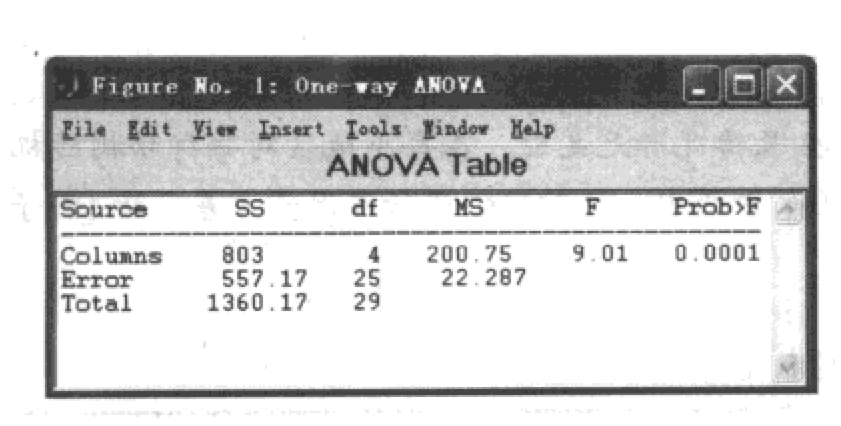


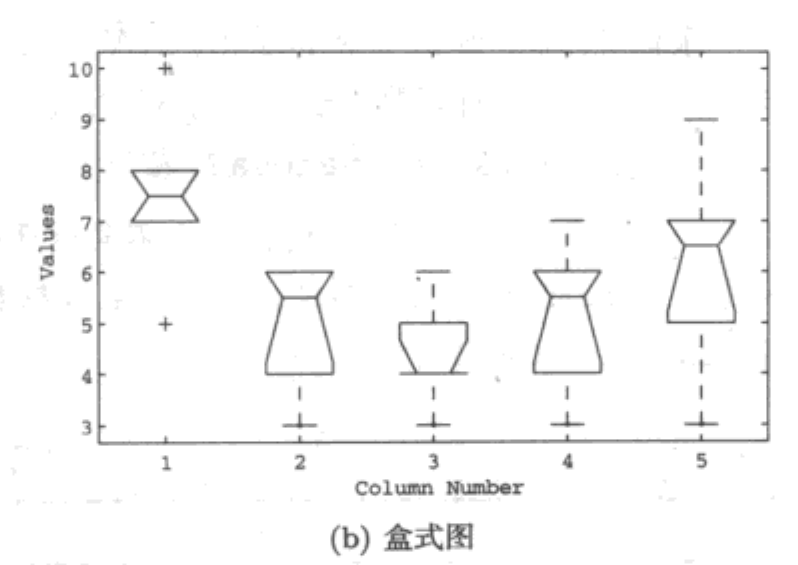
Figure No. 1: One-way ANOVA

File Edit View Insert Tools Window Help

ANOVA Table

Source	SS	df	MS	F	Prob>F
Columns	803	4	200.75	9.01	0.0001
Error	557.17	25	22.287		
Total	1360.17	29			

(a) 单因子方差表



(b) 盒式图

图 9-14 单因子方差分析结果

根据双因子的特点，可以引入 3 个假设如下：

$$\begin{cases} \mathscr{H}_1: \alpha_1=\alpha_2=\cdots=\alpha_I, & \alpha_i \text{ 为第一因子单独作用的效应} \\ \mathscr{H}_2: \beta_1=\beta_2=\cdots=\beta_J, & \beta_j \text{ 为第二因子单独作用的效应} \\ \mathscr{H}_3: \gamma_1=\gamma_2=\cdots=\gamma_{IJ}, & \gamma_k \text{ 为两个因子同时作用的效应} \end{cases} \tag{9-5-1}$$

对双因子方差分析问题，可以构造出如表 9-6 所示的分析表格。其中，交互效应的 SSAE 的值可以由式 (9-5-2) 求出。

表 9-6 双因子方差分析表

方差来源	平方和	自由度	均 方	F	p 值
主效应 A	$\text{SSA}=JK\sum_i \bar{y}_{i,:,:}^2-IJK\bar{y}_{:,:,:}^2$	$I-1$	$\text{MSSA}=\dfrac{\text{SSA}}{I-1}$	MSSA/MSSE	p_A
主效应 B	$\text{SSB}=IK\sum_i \bar{y}_{:,j,:}^2-IJK\bar{y}_{:,:,:}^2$	$J-1$	$\text{MSSB}=\dfrac{\text{SSB}}{J-1}$	MSSB/MSSE	p_B
交互效应	SSAB 见式 (9-5-2) 中的定义	$(I-1)(J-1)$	$\text{MSSAB}=\dfrac{\text{SSAB}}{(I-1)(J-1)}$	MSSAB/MSSE	p_{AB}
随机误差	$\text{SSE}=\sum_{ijk} y_{i,j,k}^2-K\sum_i\sum_j \bar{y}_{i,j,:}^2$	$IJ(K-1)$	$\text{MSSE}=\dfrac{\text{SSE}}{IJ(K-1)}$		
和	$\text{SST}=\sum_{ijk} y_{i,j,k}^2-IJK\bar{y}_{:,:,:}^2$	$IJK-1$			

$$\text{SSAB}=K\sum_{ij}\bar{y}_{i,j,:}^2-JK\sum_i\bar{y}_{i,:,:}^2-IK\sum_j\bar{y}_{:,j,:}^2+IJK\bar{y}_{:,:,:}^2 \tag{9-5-2}$$

另外，3 个概率的定义及意义为

$$\begin{cases} p_A=P\left(F_{[I-1,IJ(K-1)]}>c_1\right) & \text{若 } p_A<c_1 \text{ 则拒绝假设 } \mathscr{H}_1 \\ p_B=P\left(F_{[J-1,IJ(K-1)]}>c_2\right) & \text{若 } p_B<c_2 \text{ 则拒绝假设 } \mathscr{H}_2 \\ p_{AB}=P\left(F_{[(I-1)(J-1),IJ(K-1)]}>c_3\right) & \text{若 } p_{AB}<c_3 \text{ 则拒绝假设 } \mathscr{H}_3 \end{cases} \tag{9-5-3}$$

求解双因子方差分析问题的 MATLAB 统计学工具箱函数为 anova2()，其调用格式与单因子方差分析函数 anova1() 很相近，为

```
[p,tab,stats]=anova2(X)
```

【例 9-29】设为比较 3 种松树在 4 个不同地区的生长情况有无差别，在每个地区对每种松树随机地选择 5 株，测量它们的胸径，得出的数据在表 9-7 中给出，试对它们进行双因子方差分析 (例子及数据来源：文献 [23]①)。

表 9-7　松树数据

	地区																			
松树种类	1					2					3					4				
1	23	15	26	13	21	25	20	21	16	18	21	17	16	24	27	14	17	19	20	24
2	28	22	25	19	26	30	26	26	20	28	19	24	19	25	29	17	21	18	26	23
3	18	10	12	22	13	15	21	22	14	12	23	25	19	13	22	16	12	23	22	19

【求解】按下面的方式将表中数据输入到 MATLAB 环境，然后调用 anova2() 函数，将得出如图 9-15 所示的结果。该结果与文献 [23] 中的结果完全一致。

```
>> B=[23,15,26,13,21,25,20,21,16,18,21,17,16,24,27,14,17,19,20,24;
      28,22,25,19,26,30,26,26,20,28,19,24,19,25,29,17,21,18,26,23;
      18,10,12,22,13,15,21,22,14,12,23,25,19,13,22,16,12,23,22,19];
   anova2(B',5);
```

Figure No. 5: Two-way ANOVA

File　Edit　View　Insert　Tools　Window　Help

ANOVA Table

Source	SS	df	MS	F	Prob>F
Columns	355.6	2	177.8	9.68	0.0003
Rows	49.65	3	16.55	0.9	0.4478
Interaction	106.4	6	17.733	0.97	0.4588
Error	882	48	18.375		
Total	1393.65	59			

图 9-15　本例子的双因子方差分析结果

从得出的结果看，由于 p_A 的值很小，所以应该拒绝 $\mathscr{H}_1$ 假设。可以认为，A 因子对观测现象有显著影响，得出结论为树种对观测树的胸径有显著影响。结合下面计算出的各个均值为

```
>> C=[];
   for i=1:3
      for j=1:4
         C(i,j)=mean(B(i,[1:5]+(j-1)*5));
   end, end
```

①第 3 种树在第 4 地区的第一个数值 16，原数据为 18，但和后面分析结果对不上，故改。

```
C=[C; mean(C)];  C=[C mean(C')']
C =
 19.6000000000  20.0000000000  21.0000000000  18.8000000000  19.8500000000
 24.0000000000  26.0000000000  23.2000000000  21.0000000000  23.5500000000
 15.0000000000  16.8000000000  20.4000000000  18.4000000000  17.6500000000
 19.5333333334  20.9333333333  21.5333333333  19.4000000000  20.3500000000
```

可见，树种 2 的树胸径最大，树种 3 的最小 (见 C 矩阵的各列)。由于另外两个概率 p_B 和 p_{AB} 的值很大，所以没有理由拒绝另外两个假设。故得出结论：地区对树的胸径无显著影响，不同区域对不同树种的胸径观测结果也无显著影响。

9.5.3 多因子方差分析

类似于前面介绍的双因子方差分析，用 MATLAB 语言的统计学工具箱还可以进行三因子甚至多因子的方差分析，可以采用 manova1() 函数进行多因子方差分析，这里不再介绍该分析。

9.6 本章要点简介

- 本章介绍的概率、统计分析的函数由下表给出。

函数名	函数功能	工具箱	本书页码
normpdf()	正态分布的概率密度函数，类似的还有 normcdf()，norminv() 和 normrnd() 函数，可以分别求出概率分布函数、逆概率分布函数及正态分布伪随机数生成函数	统计学	301,308
gampdf()	Γ 分布的概率密度函数，类似的还有 gamcdf()，gaminv() 和 gamrnd() 函数，可以分别求出概率分布函数、逆概率分布函数及 Γ 分布伪随机数生成函数	统计学	302,308
chi2pdf()	χ^2 分布的概率密度函数，类似的还有 chi2cdf()，chi2inv() 和 chi2rnd() 函数，可以分别求出概率分布函数、逆概率分布函数及 χ^2 分布伪随机数生成函数	统计学	303,308
tpdf()	T 分布的概率密度函数，类似的还有 tcdf()，tinv() 和 trnd() 函数，可分别求出概率分布函数、逆概率分布函数及 T 分布伪随机数生成函数	统计学	304,308
fpdf()	F 分布的概率密度函数，类似的还有 fcdf()，finv() 和 frnd() 函数，可分别求出概率分布函数、逆概率分布函数及 F 分布伪随机数生成函数	统计学	306,308
raylpdf()	Rayleigh 分布的概率密度函数，类似的还有 raylcdf()，raylinv() 和 raylrnd() 函数，可以分别求出概率分布函数、逆概率分布函数及 Rayleigh 分布伪随机数生成函数	统计学	305,308
poisspdf()	Poisson 分布的概率密度函数，类似的还有 poisscdf()，poissinv() 和 poissrnd() 函数，可以分别求出概率分布函数、逆概率分布函数及 Poisson 分布伪随机数生成函数	统计学	300,308
mean()	求取向量的均值，类似的还有 cov() 求方差，std() 求标准差	MATLAB	309

续表

函数名	函 数 功 能	工具箱	本书页码
gamstat()	求取 Γ 分布的均值和方差，类似的函数还有 normstat(), normstat(), gamstat(), raylstat() 等	统计学	310
moment()	求取高阶中心矩，高阶原点矩也可以通过相应语句得出	统计学	311
cov()	求取向量的协方差均值	MATLAB	312
mvnpdf()	多变量正态分布概率密度函数	统计学	313
mvnrnd()	多变量正态分布概率伪随机数生成函数	统计学	313
normfit()	正态分布均值和方差的参数估计和区间估计，类似的函数还有 gamfit(), chi2fit(), tfit(), raylfit() 等	统计学	314
语句	多变量线性回归计算语句	语句	317
regress()	多变量线性回归计算函数	统计学	317
nlfit()	非线性最小二乘参数估计	统计学	319
nlparci()	非线性最小二乘的区间估计	统计学	319
ztest()	已知方差的正态分布均值假设检验的 Z 测试方法	统计学	323
ttest()	未知方差的正态分布均值假设检验的 T 测试方法	统计学	323
jbtest()	分布正态性的 Jarque-Bera 假设检验方法	统计学	324
lillietest()	分布正态性的 Lilliefors 假设检验方法	统计学	324
kstest()	任意分布的 Kolmogorov-Smirnov 假设检验方法	统计学	326
anova1()	单因子方差分析	统计学	327
anova2()	双因子方差分析	统计学	330
manova1()	多因子方差分析	统计学	331

- MATLAB 的统计学工具箱中提供了大量函数名有规律的函数。例如，函数名前一部分为 gam 常用于表示和 Γ 分布有关的函数，这类关键词在表 9-9 中给出。函数名的后一部分为 pdf 的表示求取概率密度的函数，cdf 表示分布函数，inv 表示逆分布函数，rnd 表示随机数生成函数，stat 表示矩阵、方差估计，fit 表示参数估计。学会了这样的组合，可以立即构造出你所需的函数名。例如，对数正态分布的参数与区间估计函数显然是 logn 和 fit 组合出的名字，即 lognfit() 函数。
- 本章介绍了各种常用的概率密度函数、概率分布函数，并绘制了用户指定参数下的概率密度、分布函数曲线。读者可以自己调用这些函数绘制出任意参数的曲线。
- MATLAB 的统计学工具箱提供了一组函数，可以按照指定的分布生成伪随机数。这些函数的名称也可以根据表 9-9 查出，可以用这些函数生成伪随机数。
- 给出了一些常用统计量的数学定义和 MATLAB 语言求解方法及函数，如均值、方差，原点矩与中心矩、协方差矩阵等，还介绍了多变量分布伪随机数生成方法。
- 介绍了参数估计与区间估计问题的 MATLAB 语言求解，并对线性多元回归和一般非线性回归问题介绍了参数与区间估计算法。

表 9-9 统计学工具箱中函数名关键词一览表

关键词	分布名称	有关参数	关键词	分布名称	有关参数	关键词	分布名称	有关参数
beta	β 分布	a,b	bino	二项分布	n,p	chi2	χ^2 分布	k
ev	极值分布	μ,σ	exp	指数分布	λ	f	F 分布	p,q
gam	Γ 分布	a,λ	geo	几何分布	p	hyge	超几何分布	m,p,n
logn	对数正态分布	μ,σ	mvn	多变量正态分布	$\boldsymbol{\mu},\boldsymbol{\sigma}$	nbin	负二项分布	ν_1,ν_2,δ
ncf	非零 F 分布	k,δ	nct	非零 T 分布	k,δ	ncx2	非零 χ^2 分布	k,δ
norm	正态分布	μ,σ	poiss	Poisson 分布	λ	rayl	Rayleigh 分布	b
t	T 分布	k	unif	均匀分布	a,b	wbl	Weibull 分布	a,b

- 介绍了假设检验的概念，并对均值假设检验、正态分布假设检验和给定分布假设检验等问题给出了基于 MATLAB 语言的求解方法。
- 介绍了方差分析问题及其 MATLAB 求解，还介绍了单因子方差分析、双因子方差分析等内容及基于 MATLAB 语言的求解方法。

9.7 习 题

1 假设已知 Rayleigh 分布的概率密度函数为 $p_r(x)=\begin{cases}\dfrac{x}{b^2}\mathrm{e}^{-\frac{x^2}{2b^2}} & x\geqslant 0\\ 0 & x<0\end{cases}$，试用解析推导的方法求出该分布的分布函数、均值、方差、中心矩和原点矩。生成一组满足 Rayleigh 分布的伪随机数，用数值方法检验得出的解析结果是否正确。

2 假设某两地 A、B 间有 6 个交通岗，在各个交通岗处遇到红灯的概率均相同，为 $p=1/3$，且中途遇到红灯的次数满足二项分布 $B(6,p)$[56]，试求出某人从 A 出发到达 B 处至少遇到一次红灯的概率。若选择不同的 p 值，试再绘制出至少遇到一次红灯的概率曲线。

3 某次外语考试抽样调查结果表明，学生外语考试成绩近似服从正态分布，且其均值为 72 分，并已知超过 96 分的人数占总数的 2.3%，试求出考生外语成绩介于 60,80 之间的概率。

4 试生成满足正态分布 $N(0.5,1.4^2)$ 的 30000 个伪随机数，对其均值和方差进行验证，并用直方图的方式观察其分布与理论值是否吻合。若改变直方图区间的宽度会得出什么结论?

5 假设通过实验测出某组数据，试用 MATLAB 对这些数据进行检验。

① 若认为该数据满足正态分布，且标准差为 1.5，请检验该均值为 0.5 的假设是否成立。

② 若未知其方差，试再检验其均值为 0.5 的假设是否成立。

③ 试对给出数据的正态性进行检验。

-1.7908	1.5803	1.5924	2.7278	-0.71772	-1.8152	2.8943	0.47035	-1.5161	0.74033	2.3831	2.3258
0.090316	2.0033	0.48874	0.99252	-2.5004	1.047	-0.052119	-0.80559	0.80414	4.6585	-1.1251	1.9318
3.9223	0.32378	-0.12149	1.0887	2.9135	-2.3273	-2.9145	5.3067	0.18716	-0.11899	-1.1234	3.4477
0.41351	2.5006	3.372	3.2303	-1.1022	-0.28116	0.5219	-0.079639	-2.1176	5.4782	0.047343	1.236
3.2618	5.6959	4.6927	-0.118	0.47461	-1.6181	0.66059	-2.6714	3.1634	3.8942	0.45396	-1.0142
-1.0665	1.6804	-0.67576	0.20045	0.49816	-2.1428	1.2122	4.4827	0.46528	-3.8764	1.1275	0.16401
0.51693	0.47348	0.73271	-2.3586	-0.061232	-1.7976	1.6246	1.2325	1.7065	-3.2812	2.8812	-5.0103
-1.2615	2.5546	-1.3172	-3.2431	1.3923	-0.40375	3.3757	-2.0178	-1.112	-0.79045	1.8988	1.5649
1.8206	0.62587	2.031	-1.083	-0.09403	0.89075	-0.73259	1.8958	-0.97501	0.081913	-3.4389	0.76313
-0.065217	-1.9909	-4.8203	1.132	-3.244	0.23873	1.0868	3.357	1.2073	0.5201	2.069	-0.82998

6 某研究者对随机抽取的一组保险丝进行了实验，测出使保险丝烧断的电流值为 10.4, 10.2, 12.0, 11.3, 10.7, 10.6, 10.9, 10.8, 10.2, 12.1 A，假设这些值满足正态分布，试在置信水平 $\alpha \leqslant 0.05$ 的条件下求出这些保险丝的溶断电流及其置信区间。

7 假设在某固定气压下对水的沸点进行多次测试，得出一组数据为 113.53, 120.25, 106.02, 101.05, 116.46, 110.33, 103.95, 109.29, 93.93, 118.67°C，并假设它们满足正态分布，试求出置信水平 $\alpha \leqslant 0.05$ 的条件下，该气压下水沸点的总体方差的置信区间。

8 甲、乙两位化验员独立地对某种聚合物的含氮量用相同的方法各取了 10 次测定，其测定值的修正样本方差 S_1^{*2}, S_2^{*2} 依次为 0.5419 和 0.6050，求总体方差比 σ_1^2/σ_2^2 置信度为 0.90 的置信区间. 假定测定值总体服从正态分布。

9 假设测出某随机变量的 12 个样本为 9.78, 9.17, 10.06, 10.14, 9.43, 10.60, 10.59, 9.98, 10.16, 10.09, 9.91, 10.36，试求其方差及方差的置信区间。

10 10 个失眠者服用 A、B 两种药后，延长睡眠时间由下表给出，试判定两种药物对失眠的疗效有无显著差异。

A	1.9	0.8	1.1	0.1	-0.1	4.4	5.5	1.6	4.6	3.4
B	0.7	-1.6	-0.2	-1.2	-0.1	3.4	3.7	0.8	0	2

11 假设两个随机变量 A,B 的样本点如下，试判定二者是否有显著差异。

A	10.42	10.48	7.98	8.52	12.16	9.74	10.78	10.18	8.73	8.88	10.89	8.1
B	12.94	12.68	11.01	11.68	10.57	9.36	13.18	11.38	12.39	12.28	12.03	10.8

12 假设测出一组输入值 x_1, x_2, x_3, x_4, x_5 和输出值 y，且已知 $y=a_1x_1+a_2x_2+a_3x_3+a_4x_4+a_5x_5$，试用线性回归方法估计出 a_i 的值及其置信区间。

x_1	8.11	9.25	7.63	7.89	12.94	10.11	7.57	9.92	7.74	7.3	9.48	11.91
x_2	2.13	2.66	0.83	1.54	1.74	0.79	0.68	2.93	2.01	1.35	2.81	2.23
x_3	-3.98	0.68	-1.42	0.96	0.28	-3.37	-4.58	-2.15	-2.66	-3.69	-1	0.98
x_4	-6.55	-6.85	-6.25	-5.34	-6.85	-7.2	-6.12	-6.07	-5.51	-6.6	-6.15	-6.43
x_5	5.92	7.54	5.39	4.65	6.47	5.1	6.04	5.37	6.54	6.55	5.8	3.95
y	27.676	38.774	23.314	23.828	35.154	21.779	25.516	29.845	32.642	28.443	31.5	23.554

13 假设测出一组输入值 x_i 和输出值 y_i，且已知原型函数为 $f(x) = a_1\mathrm{e}^{-a_2x}\cos(a_3x + \pi/3) + a_4\mathrm{e}^{-a_5x}\cos(a_6x + \pi/4)$，试估计出 a_i 的值及其置信区间。

x	1.027	1.319	1.204	0.684	0.984	0.864	0.795	0.753	1.058	0.914	1.011	0.926
y	-8.8797	-5.9644	-7.1057	-8.6905	-9.2509	-9.9224	-9.8899	-9.6364	-8.5883	-9.7277	-9.023	-9.6605

14 假设测出一组输入值 x_1, x_2, x_3, x_4, x_5 和输出值 y 如下表，且已知 $y = \mathrm{e}^{-a_1x_1}\sin(a_2x_2 + a_3) + x_4^2\cos(a_4x_5)$，试估计出 a_i 的值及其置信区间。

x_1	8.11	9.25	7.63	7.89	12.94	10.11	7.57	9.92	7.74	7.3	9.48	11.91
x_2	2.13	2.66	0.83	1.54	1.74	0.79	0.68	2.93	2.01	1.35	2.81	2.23
x_3	-3.98	0.68	-1.42	0.96	0.28	-3.37	-4.58	-2.15	-2.66	-3.69	-1	0.98
x_4	-6.55	-6.85	-6.25	-5.34	-6.85	-7.2	-6.12	-6.07	-5.51	-6.6	-6.15	-6.43
x_5	5.92	7.54	5.39	4.65	6.47	5.1	6.04	5.37	6.54	6.55	5.8	3.95
y	22.126	250.16	-144.11	-152.07	234.09	-318.04	54.401	-136.8	132.03	229.26	-19.048	-145.83

15 假设可以通过实验测出如下的数据，且假设这些数据满足 $y(t) = c_1\mathrm{e}^{-5t}\sin(c_2t) + (c_3t^2 + c_4t^3)\mathrm{e}^{-3t}$，试根据下面的数据求出 c_i 参数的估计值与置信区间。

-0.21628	0.12012	1.8787	2.7393	2.7238	4.5219	5.0833	4.9699	5.5947	5.9073	6.0663	6.8166
6.4115	8.0106	7.0286	7.2988	7.8903	7.4742	7.4594	7.1308	7.7132	6.8981	7.9065	8.3289
7.1251	7.8416	7.9701	6.4669	6.4553	7.3657	6.7779	7.2148	7.1647	6.9958	7.1645	6.7303
6.8659	5.5421	6.005	5.8074	4.9543	5.7555	4.9696	6.077	4.8393	5.3799	5.1003	4.4062
3.6602	4.5961	4.0026	4.6994	4.5325	5.0136	4.3541	3.6301	4.0379	3.2414	3.637	3.5258
3.4556	3.2048	3.8218	2.2502	3.3167	3.4682	3.306	3.1518	2.8077	3.053	2.928	2.447
2.3194	2.2955	1.6433	2.2031	2.3206	2.3618	2.871	1.9203	2.3557	2.3935	2.4159	1.4025
1.9591	1.928	1.2625	1.3541	2.2263	1.5807	1.8039	1.6166	1.2197	1.2236	1.6922	0.96335
1.7978	1.6616	0.93709	1.1868	0.69824	0.16428	1.7327	0.95505	1.3536	1.2832	1.1538	0.61867
0.62523	0.89035	0.46394	0.5088	1.7534	1.0259	0.37077	0.94067	0.37941	0.25167	0.77891	1.3697
0.94126	1.1895	0.26198	0.60059	0.68501	0.19531	0.12812	1.2397	0.76627	0.4249	1.1374	0.83772
0.21463	1.3671	0.83017	1.4132	1.2313	0.3089	-0.006057	0.59259	0.45308	0.18609	0.8465	1.3317
0.30432	0.1444	0.435	0.88024	0.11233	-0.070364	0.46137	0.47981	0.74641	0.19745	0.41194	0.2611
0.43065	1.2299	0.15113	-0.22709	0.67362	-0.020406	0.44189	0.10287	0.67713	0.67878	0.29351	-0.63869
0.44798	1.1715	0.87767	-0.4282	0.3163	0.0085371	-0.16908	-0.28006	0.47553	0.11058	0.34733	0.12978
0.075603	0.48803	0.66124	1.3478	-0.3922	-0.23006	0.79504	0.076187	-0.42451	0.41902	1.0331	0.60566

16 设从 A,B 两个不同的地区各取得某种植物的样品 12 个，测得植物中铁元素含量 (μg／g)的数据如下，假定已经知道这种植物中铁元素含量分布为正态，且分布的方差是不受地区影响的，检验这两个地区该种植物中铁元素含量的分布是否相同。

地区 A	11.5	18.6	7.6	18.2	11.4	16.5	19.2	10.1	11.2	9	14	15.3
地区 B	16.2	15.2	12.3	9.7	10.2	19.5	17	12	18	9	19	10

17 一批由同种原料织成的布，用不同的染整工艺处理，每台进行缩水串试验，目的是考察不同的工艺对布的缩水率是否有显著影响。现采用 5 种不同的染整工艺，每种工艺处理 4 块布样，测得缩水率的百分数见下表。试判定染整工艺对缩水率有无显著影响。

布样	染整工艺数据					布样	染整工艺数据				
1	4.3	6.1	6.5	9.3	9.5	2	7.8	7.3	8.3	8.7	8.8
3	3.2	4.2	8.6	7.2	11.4	4	6.5	4.2	8.2	10.1	7.8

18 抽查某地区 3 所小学五年级男学生的身高由下表给出，问该地区这 3 所小学五年级男学生的平均身高是否有显著差别 ($\alpha = 0.05$)?

学校	实测身高数据					
1	128.1	134.1	133.1	138.9	140.8	127.4
2	150.3	147.9	136.8	126	150.7	155.8
3	140.6	143.1	144.5	143.7	148.5	146.4

19 下表记录了 3 位操作工分别在 4 台不同机器上操作的日产量，试检验

① 操作工之间的差异是否显著?

② 机器之间的差异是否显著?

③ 交互作用是否显著 ($\alpha = 0.05$)?

机器	操作工									机器	操作工								
	1			2			3				1			2			3		
M_1	15	15	17	19	19	16	16	18	21	M_3	15	17	16	18	17	16	18	18	18
M_2	17	17	17	15	15	15	19	22	22	M_4	18	20	22	15	16	17	17	17	17

第10章　数学问题的非传统解法

前面各章系统介绍了高等应用数学各个领域的数学问题计算机辅助求解方法。近几十年来，科学家们仿照人类思维方式或其他自然科学的研究成果，发展出了很多新的分支，用来解决数学和其他应用科学领域的问题。例如，仿照人类思维和语言规则提出的模糊逻辑和模糊推理，仿照生物神经网络提出的人工神经网络，仿照生物遗传学及进化过程的“适者生存”规律提出的遗传算法和进化理论等。这些理论在自动控制学科及其他科学与工程领域均有很好的应用前景。本书将在第10.1节中首先介绍经典集合论问题的MATLAB语言求解方法，然后引入模糊集合的概念并介绍基于MATLAB语言的模糊集合与模糊推理的实现方法。第10.2节引入人工神经网络的数学表示及反馈时神经网络结构，介绍利用用MATLAB语言神经网络结构设置、训练及网络泛化的全过程，利用MATLAB神经网络工具箱直接求解数拟合问题的方法。第10.3节实现引入遗传算法的基本概念和解题步骤，并介绍其在无约束最优化与有约束最优化问题中的应用，并通过例子介绍利用MATLAB语言现成的工具求解最优化问题的方法。第10.4节介绍小波理论和小波分析概述，并介绍如何用MATLAB语言的小波工具箱求解噪声滤波等。第10.5节将介绍粗糙集的基本理论，然后介绍其在条件约简等领域的应用。第10.6节还将介绍分数阶微积分及微分方程的近似方法。本章简要介绍这些理论的基本概念，但均侧重于介绍用MATLAB语言或相应的工具箱如何求解这些问题的方法介绍。

10.1　模糊逻辑与模糊推理

10.1.1　经典集合论和模糊集

10.1.1.1　经典可枚举集合论问题及MATLAB求解

集合论是现代数学的基础。所谓集合，就是一些事物的全体，而其中每一个事物均称为集合中的一个元素。若事物a是集合A中的一个元素，则记$a \in A$，称为a属于A。若b不是A集合中的元素，则记$b \notin A$。所谓可枚举集合，就是该集合中的所有元素均可以一一列出的集合。在MATLAB中用向量或单元数组的形式就可以表示这样的集合。例如，下面的语句均可以表示集合。

```
>> A=[1 2 3 5 6 7 9 3 4 11]  % 数字构成的集合，可以有重复元素
   B={1 2 3 5 6 7 9 3 4 11}  % 上述集合的单元数组表示方法，二者等价
   C={'ssa','jsjhs','su','whi','kjshd','kshk'}  % 字符串集合，可以为人名等
```

MATLAB语言提供了集合定义与基本运算函数。在表10-1中列出了进行集合运算的函数及解释，用这些函数可以对集合进行操作，这些函数还可以嵌套使用，建立较复杂的集合运算。遗憾的是，这些函数不能用于符号表达式的集合运算。

表 10-1 MATLAB 下集合运算的函数

运算名称	MATLAB 语句	集合运算描述
并集运算	A=union(B,C)	求两个集合 B 和 C 的并集，数学记号为 $A=B\bigcup C$，集合运算后的结果被重新排序
差集运算	A=setfiff(B,C)	求两个集合 B 和 C 的差集，记作 $A=B\backslash C$，亦即从集合 B 中剔除 C 中的元素剩下的元素，结果被重新排序
交集运算	A=intersect(B,C)	求两个集合 B 和 C 的交集，即 $A=B\bigcap C$，重新排序
异或运算	A=setxor(B,C)	求两个集合 B 和 C 的异或运算，即从 $B\bigcup C$ 中剔除 $B\bigcap C$，数学表示为 $A=(B\bigcup C)\backslash(B\bigcap C)$，结果被重新排序
惟一运算	A=unique(B)	将 B 集合中的重复元素剔除，得出的是惟一的元素集合，结果被排序
属于判定	key=ismember(a,B)	判定 a 是否为 B 集合中的元素，如果是则返回 key 值为 1，否则返回 0，记作 key=$a\in B$。其实，在属于关系中，a 也可以为矩阵，这时返回的 key 为和 a 一样维数的矩阵，在满足属于关系的元素处为 1，否则为 0

【例 10-1】假设给定 3 个集合 $A=\{1,4,5,8,7,3\}$，$B=\{2,4,6,8,10\}$，$C=\{1,7,4,2,7,9,8\}$，试演示集合的各种运算，并验证这些集合满足交换律 $(A\bigcup B)\bigcap C=(A\bigcap C)\bigcup(B\bigcap C)$。

【求解】由给出的条件可以立即输入已知的 A,B,C 这 3 个集合，然后调用集合运算的命令即可以得出如下的结果，这些结果应该不难理解。

```
>> A=[1,4,5,8,7,3]; B=[2,4,6,8,10]; C=[1,7,4,2,7,9,8];   % 集合定义
   D=unique(C)  % 求解惟一运算，可见从 C 中剔除了重复的 7
D =
     1     2     4     7     8     9
>> E=union(A,B)   % 求出并集
E =
   1     2     3     4     5     6     7     8     10
>> F=intersect(A,B)   % 求出交集
F =
     4     8
```

给出如下命令，则可以发现交换律左侧的集合与右侧的集合求差集，得出的结果为空集，由此验证了交换律的正确性。

```
>> G=setdiff(intersect(union(A,B),C),union(intersect(A,C),intersect(B,C)))
G =
    Empty matrix: 1-by-0
```

现在可以演示 ismember() 函数在集合运算中的应用。

```
>> ismember(A,B)
ans =
     0     1     0     1     0     0
```

由得出的结果看，A 集合中的第 2 和第 4 元素属于 B 集合，因为这些位置处的测试结果的值为 1。所以，可以用下面的语句提取出 A 集合中属于 B 的元素，亦即得出 $A\bigcap B$。

```
>> A(ismember(A,B))
ans =
     4     8
```

【例 10-2】假设 A 集合为字符串组{'skhsak','ssd','ssfa'}，B 集合{'sdsd','ssd','sssf'}，试求它们的并集与交集，令 C={'jsg','sjjfs','ssd'}，试验证结合律

$$\left(A\bigcap B\right)\bigcup\left(C\bigcap B\right)=\left(A\bigcup C\right)\bigcap B$$

【求解】字符串构成的集合可以用单元数组的形式表示，也可以进行集合运算，所以直接用下面的语句求出它们的交集和并集为

```
>> A={'skhsak','ssd','ssfa'}; B={'sdsd','ssd','sssf'}; F=union(A,B)  % 并集
F =
    'sdsd'    'skhsak'    'ssd'    'ssf'    'sssf'
>> D=intersect(A,B)   % 交集
D =
    'ssd'
>> C={'jsg','sjjfs','ssd'};  % 可以由下面的集合运算验证结合律
   E=setdiff(union(intersect(A,B),intersect(C,B)),intersect(union(A,C),B))
E =
     {}
```

子集与集合包含等概念是集合论中很重要的概念。所谓集合包含即集合 A 中所有的元素均为集合 B 的元素，记作 $A\subseteq B$，称为 B 包含 A，又称 A 是 B 的子集。若 $B\backslash A$ 非空，则称严格包含，记作 $A\subset B$。MATLAB 中并未直接提供集合包含或子集的函数，但可以通过下面的命令判定包含和严格包含。

> key=all(ismember(A,B))，key=1 则 $A\subseteq B$，即判定 A 所有元素均为 B 元素
> key=all(ismember(A,B))&(length(setdiff(B,A))>0)，key=1 则 $A\subset B$

【例 10-3】考虑例 10-2 中的 E, F 集合，试判定 $F\subset E$ 是否满足，并由 A 集合验证集合的自反律，亦即 $A\subseteq A$。

【求解】可以用下面的语句进行判定：

```
>> E=union(A,B); F=intersect(A,B); key=all(ismember(F,E))
key =
     1
```

事实上，$F=A\bigcup B$，$E=A\bigcap B$，所以当然 $E\subset F$。还可以验证 $A\subseteq A$，亦即自反律。

```
>> key=all(ismember(A,A)) & (length(setdiff(A,A))>0); % A⊄A
   key1=all(ismember(A,A)); [key,key1]  % A⊆A 当然成立
```

```
ans =

     1     0
```

10.1.1.2 模糊集合

由经典集合论可见，一个事物 a 要么就属于集合 A，要么就不属于集合 A，没有其他的属于关系。在现代科学与工程应用中，经常会出现模糊的概念，亦即某一事物 a 以一定程度属于集合 A，该思想是模糊集合的基础。

模糊集合的概念是控制论专家 Lotfi A Zadeh 教授于 1965 年引入的[53]。目前模糊逻辑已经广泛地应用于理、工、农、医各种各样的领域[44]。在自动控制领域中模糊控制也是很有吸引力的研究方向。

在本书前面的介绍中实际上也使用了模糊的概念，例如变步长方法中关于误差的描述是当“误差较大时……”，只不过在实际处理时没有使用模糊的方法去处理，而直接使用了确定性方法解决问题。

这里不加解释和翻译地直接引入文献 [24] 给出的示意图来表示精确性与意义性，如图 10-1 所示。可以看出，现实世界中的事物并非都是越精确越好。Zadeh 教授指出，当问题的复杂性增加时，精确的描述将失去意义，而有意义的描述将失去精度。

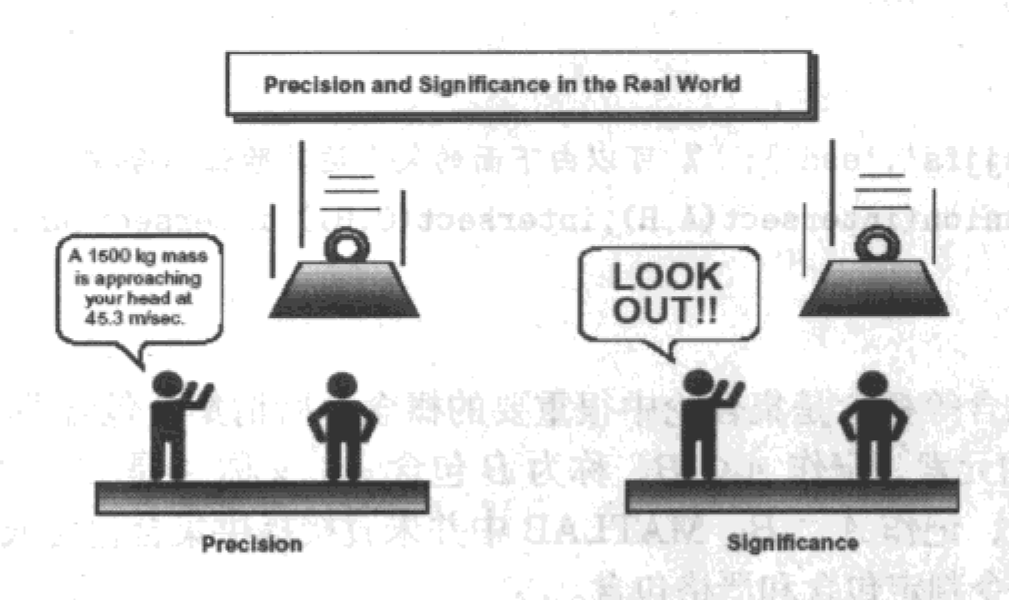


图 10-1 在现实世界中精确性与意义性示意图 (文献 [24])

10.1.2 隶属度与模糊化

10.1.2.1 钟形隶属度函数

钟形隶属函数的数学表达式为

$$f(x)=\frac{1}{1+\left|\dfrac{x-c}{a}\right|^{2b}} \tag{10-1-1}$$

MATLAB 模糊逻辑工具箱中提供了函数 `gbellmf()`，可以求出隶属度函数的值。该函数的调用格式为

```
y=gbellmf(x,[a,b,c])
```

其中，$\boldsymbol{x}$ 为任意给定的自变量值。调用此函数则可以求出 $\boldsymbol{x}$ 处的隶属度函数值 $\boldsymbol{y}$。

【例 10-4】可以用绘制出不同参数组合下的钟形隶属度函数曲线。具体的方法是，先选定 x 向量，则分别改变 a,b,c 的值，可以得出如图 10-2 所示的隶属度函数曲线，从得出的曲线可以观察出隶属度函数对 a,b,c 参数的依赖关系。

```
>> x=[0:0.05:10]'; y=[]; a0=1:5; b=2; c=3;
   for a=a0, y=[y gbellmf(x,[a,b,c])]; end
   y1=[]; a=1; b0=1:4; c=3; for b=b0, y1=[y1 gbellmf(x,[a,b,c])]; end
   y2=[]; a=2; b=2; c0=1:4; for c=c0, y2=[y2 gbellmf(x,[a,b,c])]; end
   plot(x,y); figure; plot(x,y1); figure; plot(x,y2)
```

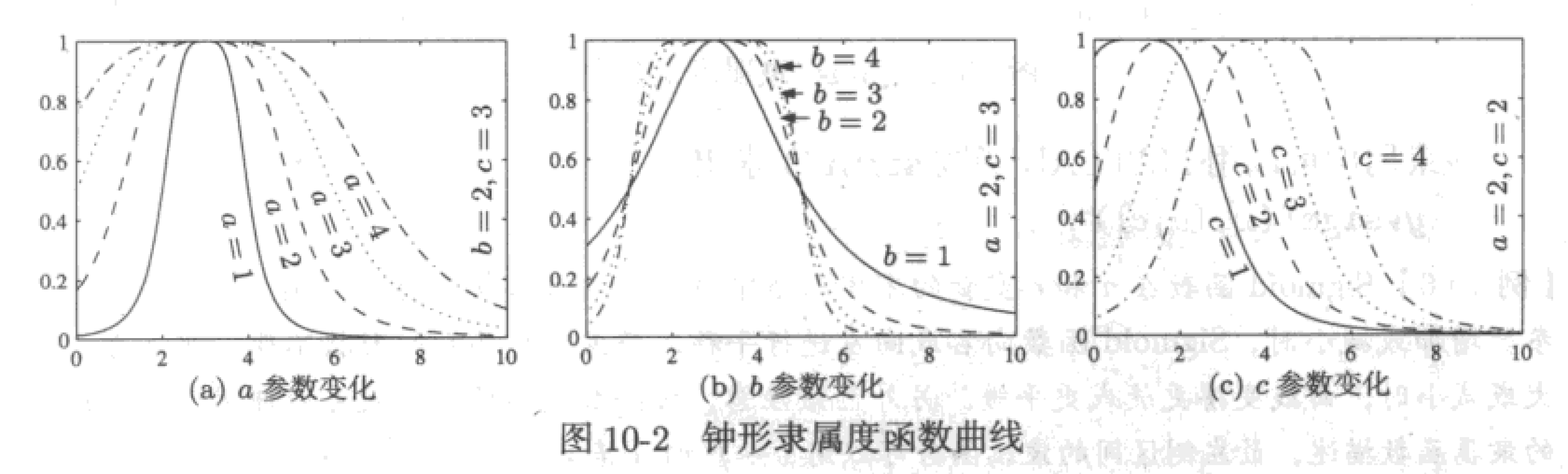


图 10-2 钟形隶属度函数曲线

从得出的曲线形状可以看出，当其他参数不变，只修改 a 值时，若 a 值小则曲线形状很窄，增大 a 的值则曲线变宽，c 参数只能用于平移曲线，不改变曲线的形状，b 参数增大将增加上升段和下降段的陡度，可以通过这些参数的组合有意识地得出合适的隶属度函数。

10.1.2.2 Gauss 隶属度函数

Gauss 隶属度函数的数学表达式为

$$f(x) = \mathrm{e}^{-\frac{(x-c)^2}{2\sigma^2}} \tag{10-1-2}$$

MATLAB 模糊逻辑工具箱中提供了 **gaussmf()** 函数，可以求取 Gauss 隶属度的值。该函数的调用格式为

y=gaussmf(x,[σ,c])

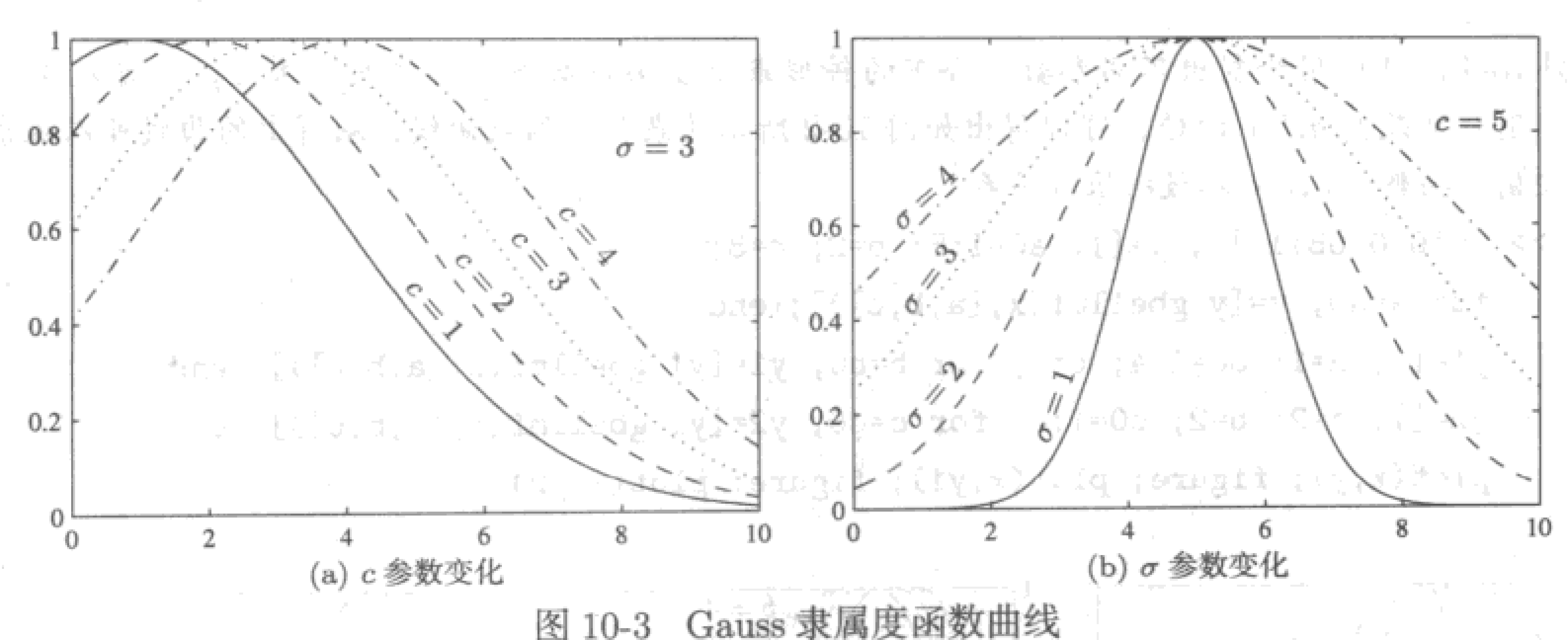
【例 10-5】不同 c 和 σ 参数的 Gauss 隶属度函数可以通过下面的语句绘制出来，如图 10-3 所示。该函数实际上和第 9 章定义的正态分布概率密度函数形状是一致的。可以看出，当 c 变化时，隶属度函数曲线形状不变，只作左右平移，σ 增大时曲线变宽。

```
>> x=[0:0.05:10]'; y=[]; c0=1:4; s=3; for c=c0, y=[y gaussmf(x,[s,c])]; end
   y1=[]; c=5; sig0=1:4; for sig=sig0, y1=[y1 gaussmf(x,[sig,c])]; end;
   plot(x,y); figure; plot(x,y1)
```

10.1.2.3 Sigmoid 型隶属度函数

Sigmoid 型隶属度函数的数学表达式为

$$f(x) = \frac{1}{1+\mathrm{e}^{-a(x-c)}} \tag{10-1-3}$$



(a) c 参数变化　(b) σ 参数变化

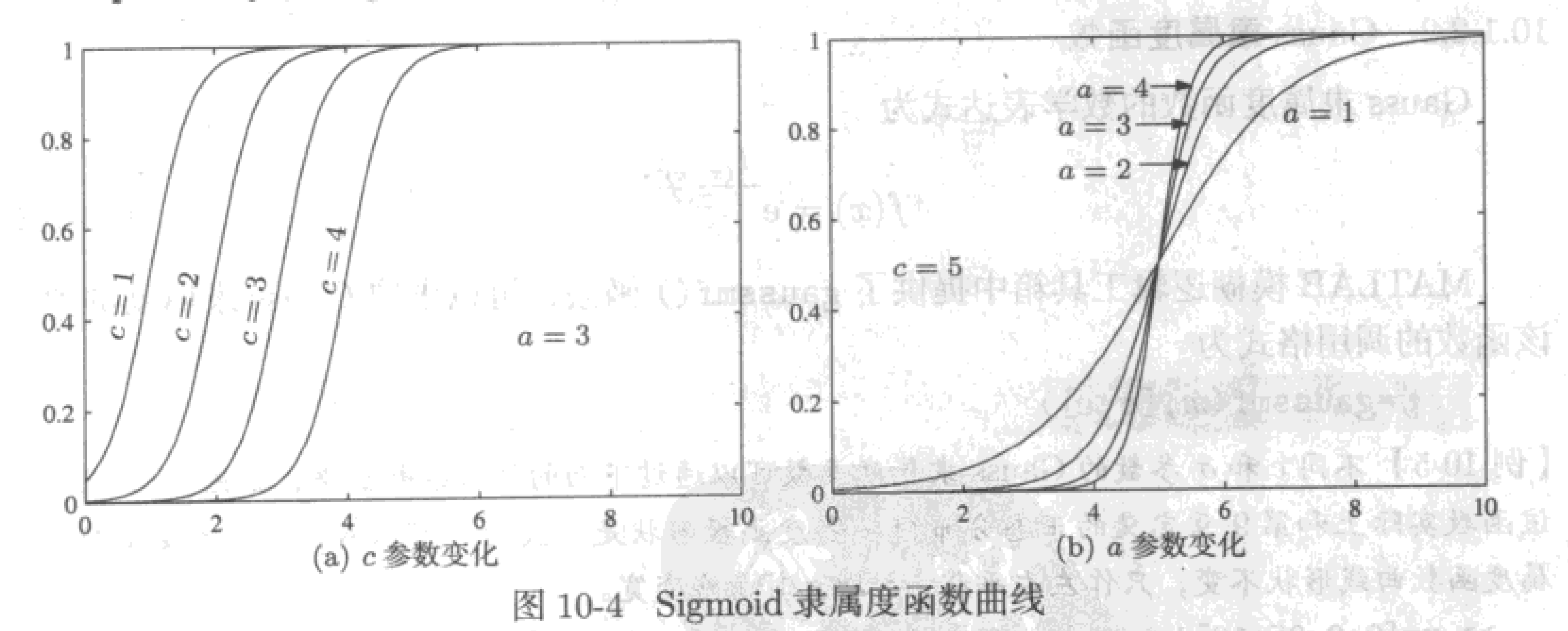
图 10-3 Gauss 隶属度函数曲线

该隶属度可以用 MATLAB 函数 sigmf() 求出

```
y=sigmf(x,[a,c])
```

【例 10-6】Sigmoid 函数在 a 和 c 变量的不同取值下隶属度函数形状如图 10-4 所示。可见，当 c 参数增加或减小时，Sigmoid 函数向右或向左进行平移，而隶属度函数的形状不变，当 a 参数增大或减小时，曲线变得更陡或更平缓。另外应该注意，该函数是单值的，故可以使用于最右区间的隶属函数描述，最左侧区间的隶属函数可以用 $1-f(x)$ 来表示。

```
>> x=[0:0.05:10]'; y=[]; c0=1:4; a=3; for c=c0, y=[y sigmf(x,[a,c])]; end
   y1=[]; c=5; a0=1:2:7; for a=a0, y1=[y1 sigmf(x,[a,c])]; end;
   plot(x,y); figure; plot(x,y1)
```



(a) c 参数变化　(b) a 参数变化

图 10-4 Sigmoid 隶属度函数曲线

10.1.2.4 隶属度函数的图形编辑界面

隶属度函数可以由 MATLAB 模糊逻辑工具箱中提供了隶属函数的编辑界面。在 MATLAB 提示符下键入 mfedit 命令就可以打开隶属函数编辑界面，如图 10-5 所示。其中给出了 3 个隶属函数的原型，用户可以通过界面中的选项设置各种隶属函数，可以由对话框右下栏目中的内容对当前隶属函数的形状和参数进行编辑，也可以通过通过鼠标在隶属函数示意图上可视地修改隶属函数的参数。

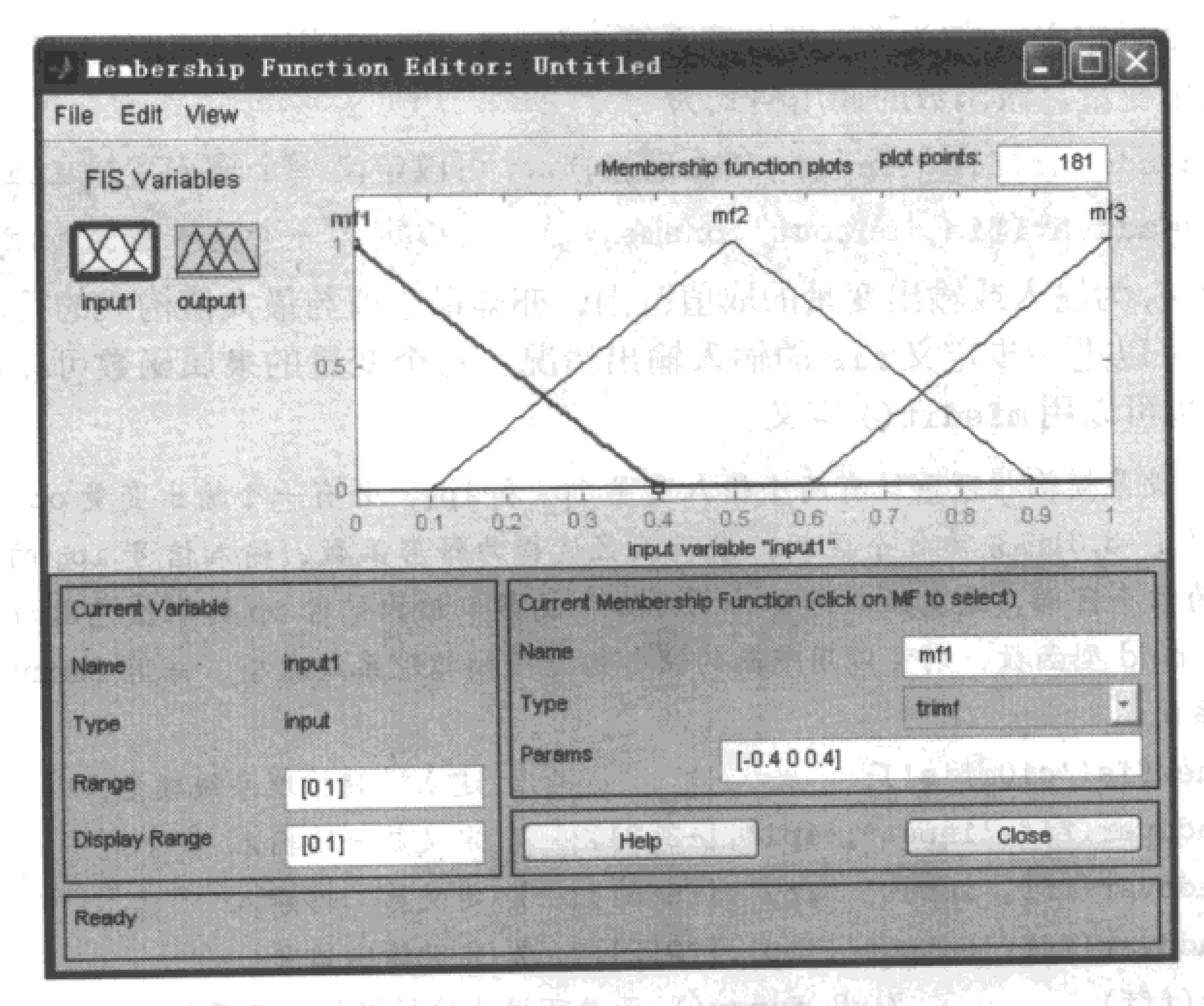
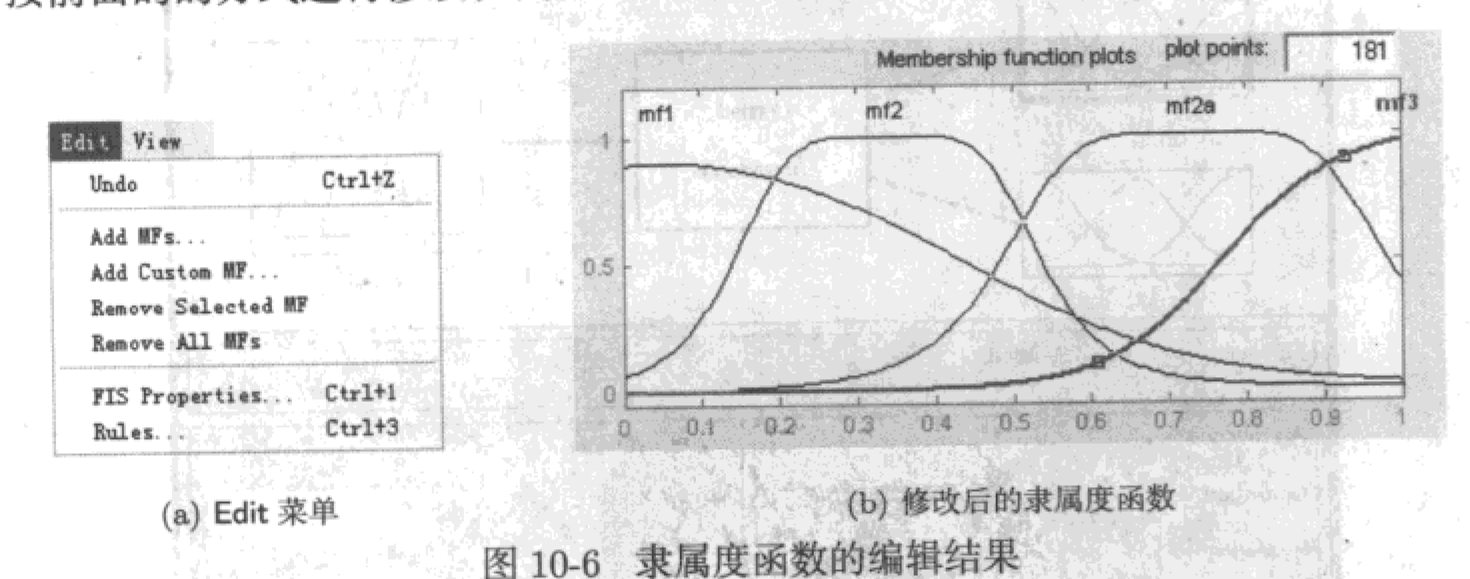


图 10-5 隶属度函数图形编辑界面

如果想再添加一个隶属函数，则可以选择 Edit → Add custom MF 菜单，如图 10-6 (a) 所示，设置完成后就可以在编辑区域能添加一个隶属函数，对这个新添加的隶属函数可以按前面的的方式进行修改，例如可以改变成如图 10-6 (b) 所示的形式。



(a) Edit 菜单　　(b) 修改后的隶属度函数

图 10-6 隶属度函数的编辑结果

10.1.3 模糊推理系统建立

用模糊逻辑工具箱中提供的 `newfis()` 函数可以构建出模糊推理系统的数据结构。其中，FIS 为模糊推理系统 fuzzy inference system 的缩写。该函数的调用格式为

```
fis=newfis(name)
```

其中，`name` 为字符串，表示模糊推理系统的名称，通过该函数可以建立起结构体 `fis`，其内容包括模糊的与、或运算，解模糊算法等，这些属性可以由 `newfis()` 函数直接定

义，也可以事后定义。定义了模糊推理系统 fis 后，可以调用 addvar() 函数来添加系统的输入和输出变量。该函数的调用格式为

fis=addvar(fis,'input',iname,v_i)　　可以定义一个输入变量 iname

fis=addvar(fis,'output',oname,v_o)　　可以定义一个输入变量 oname

其中，v_i 及 v_o 为输入或输出变量的取值范围，亦即最小值与最大值构成的行向量。通过这样的方法可以进一步定义 fis 的输入输出情况，每个变量的隶属函数可以用 addmf() 函数定义，也可以用 mfedit() 定义。

【例 10-7】假设某模糊推理系统有两个输入变量 ip_1 和 ip_2，并有一个输出变量 op，且假设 ip_1 的取值范围为 $(-3,3)$，分为 3 个区间，隶属函数选择为钟形函数，输入信号 ip_2 的取值范围为 $(-5,5)$，分为 3 个区间，隶属函数选择为 Gauss 型函数，输出信号 op 的取值范围为 $(-2,2)$，隶属函数为 Sigmoid 型函数，则可以用下面的语句构造模糊推理系统原型，并用 fuzzy() 函数编辑此模糊推理系统。

```
>> fff=newfis('c10mfis');                   % 建立模糊推理系统模型
   fff=addvar(fff,'input','ip1',[-3,3]);  % 定义第一路输入
   fff=addvar(fff,'input','ip2',[-5,5]);  % 定义第二路输入
   fff=addvar(fff,'output','op',[-2,2]);  % 定义输出信号
   fuzzy(fff)            % 用 fuzzy() 函数可视地编辑模糊推理系统
```

由 fuzzy() 函数可以打开模糊推理系统的程序界面，如图 10-7 所示。

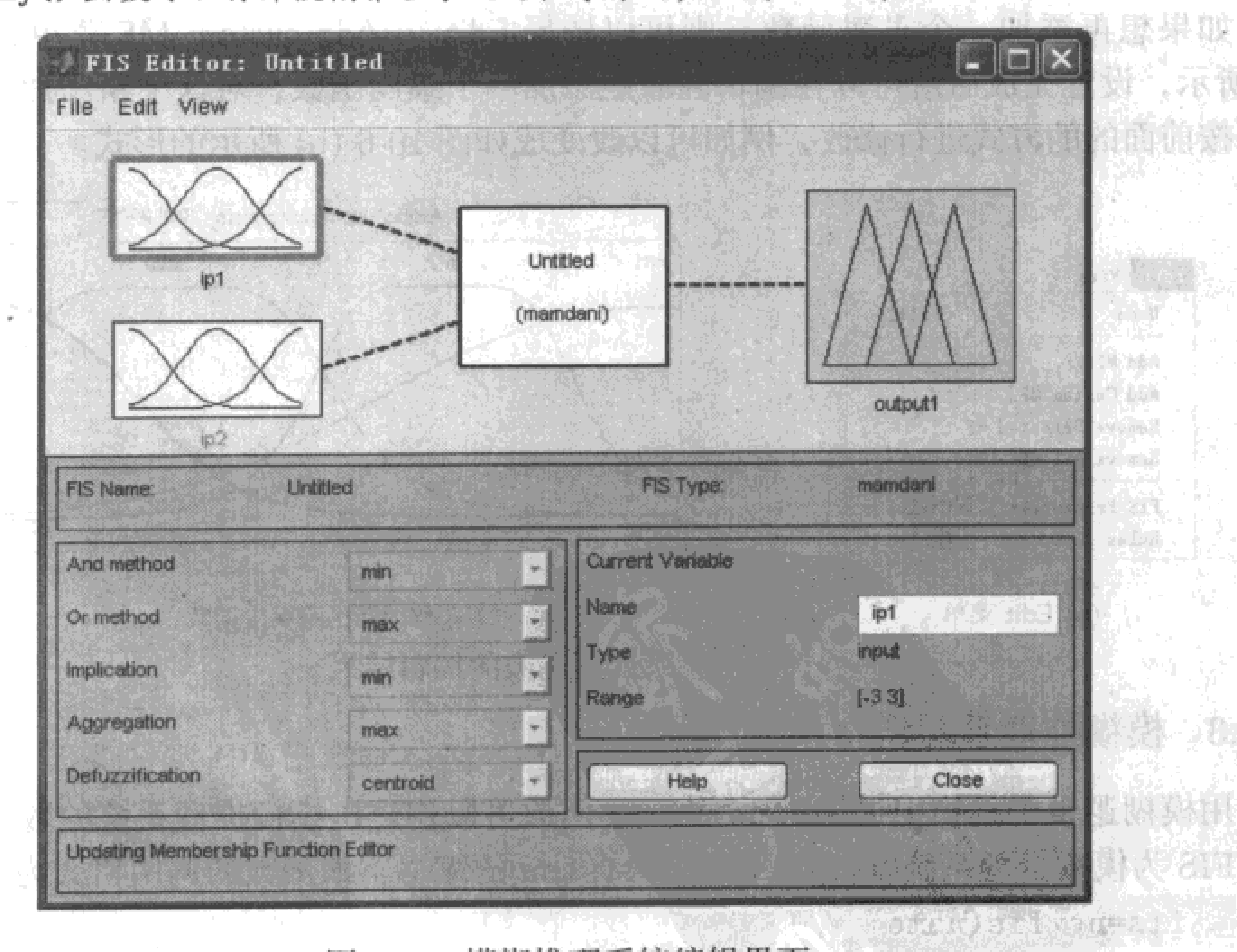
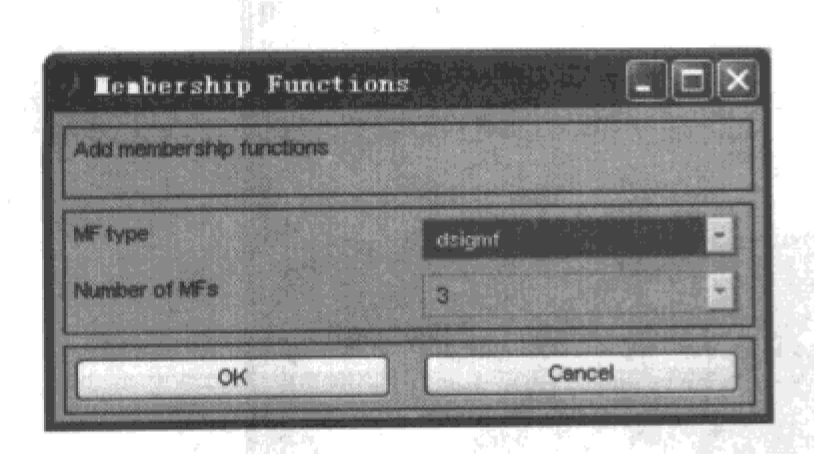


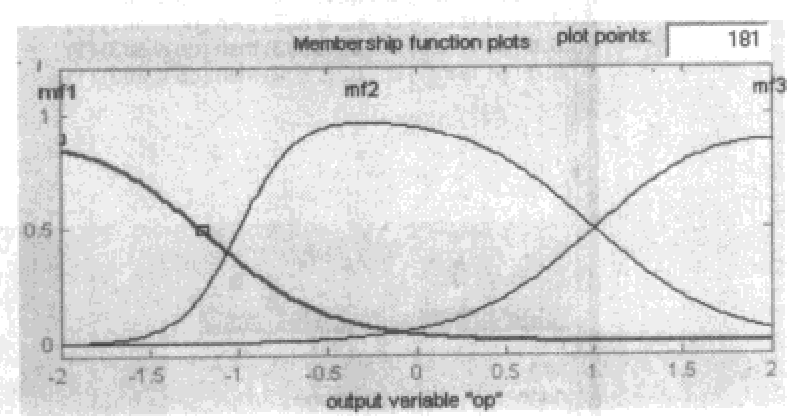
图 10-7　模糊推理系统编辑界面

在得出的界面下，选择 Edit → Membership functions 菜单项，可以打开如图 10-5 所示的隶属

度编辑界面。在得出的界面上选择 ip_1 图标，再选择 Edit → Add MFs 菜单项，可以打开如图 10-8 (a) 所示的对话框，可以通过该对话框定义各个信号的隶属函数。例如，可以通过编辑得出如图 10-8 (b) 所示的输出隶属函数。



(a) 隶属函数设置对话框



(b) 修改后的输出变量隶属度函数

图 10-8 隶属度函数的编辑结果

10.1.4 模糊规则与模糊推理

10.1.4.1 模糊化

若将某信号用三个隶属函数表示，则一般对应的物理意义是“很小”、“中等”与“较大”，若分为 5 段，则可以表示为“很小”、“较小”、“中等”、“较大”和“很大”，一个精确的信号可以通过这样一组隶属函数模糊化，变成模糊信号。

10.1.4.2 模糊规则

如果将多路信号均模糊化，则可以用 `if`，`else` 型语句表示出模糊推理关系。例如，若输入信号 ip_1“很小”，且输入信号 ip_2“较大”，则设置“较大”的输出信号 op，这样的推理关系可以表示成

```
if ip1 为“很小” and ip2 为“很大”, then op=“很大”
```

模糊推理规则可以通过 `ruleedit()` 函数生成的界面来设定，也可以从 `mfedit()` 函数界面的 Edit → Rules 菜单项编辑模糊推理规则，这将打开如图 10-9 所示的对话框。用该对话框可以逐条将推理规则输入到系统中，每设定一条规则后，可以单击 Add rule 按钮，将规则添加到规则库中。如果想删除某条规则，则选中该规则，然后单击 Delete rule 按钮即可删除。

编辑后的规则还可以单击 Change Rule 按钮进行修改。完成了模糊规则的编辑，则可以单击 Close 按钮关闭编辑窗口。模糊推理规则可以由 View → Surface 菜单项进行处理，得出如图 10-10 (a) 所示的三维图形，表明从输入信号到输出信号的映射关系。

模糊规则还可以更简单地用数据向量表示，多行向量可以构成多条模糊规则矩阵。每行向量有 $m+n+2$ 个元素，m,n 分别为输入变量和输出变量的个数，其中前 m 个元素表示输入信号的隶属函数序号，次 n 个元素对应输出信号的隶属函数序号，第 $m+n+1$ 表示输出的加权系数，最后一个元素表示输入信号的逻辑关系，1 表示逻辑“与”，2 表示逻辑“或”。例如对图 10-9 中的第 3 条逻辑关系来说，若用数据向量的形式可以表示为

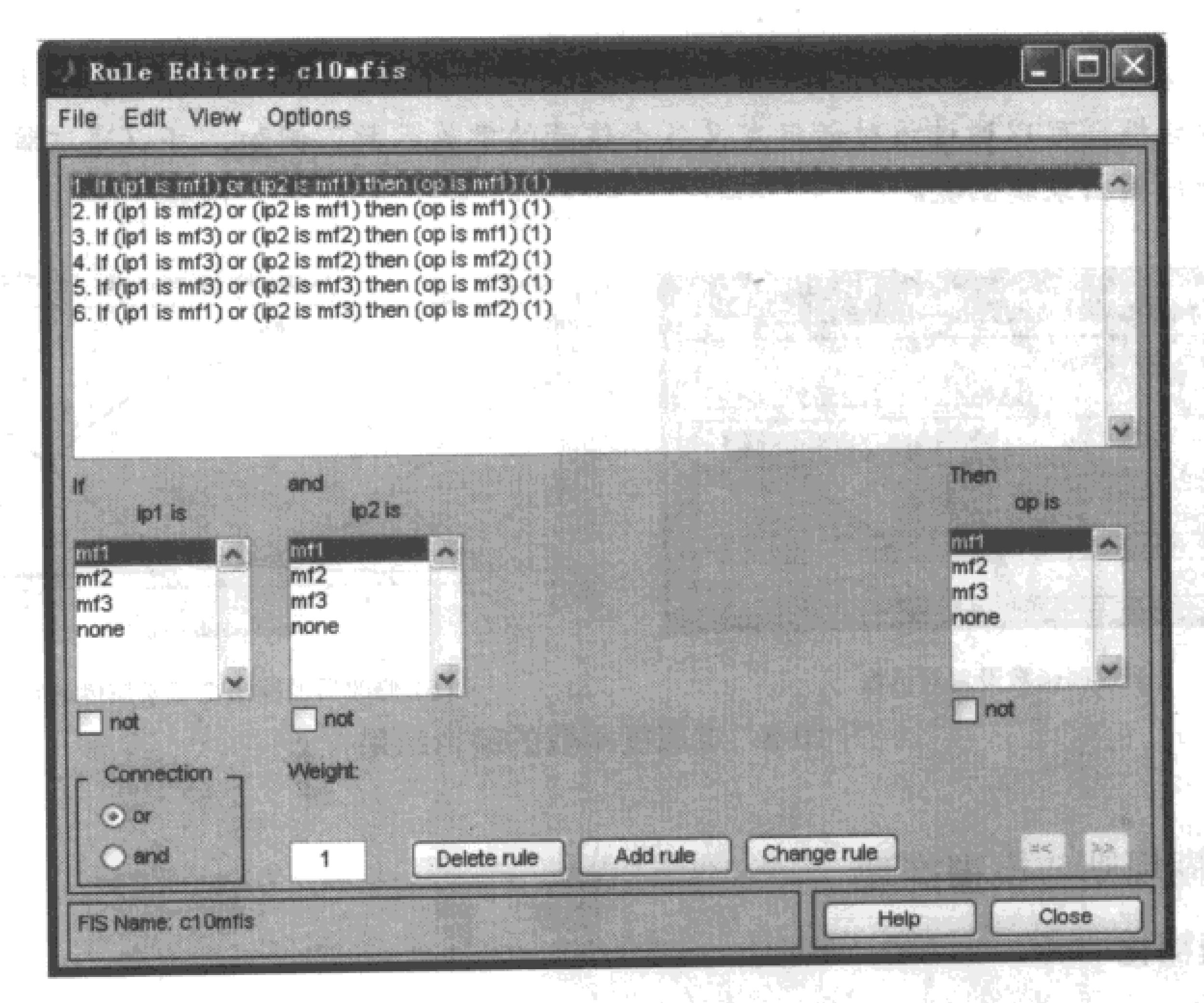
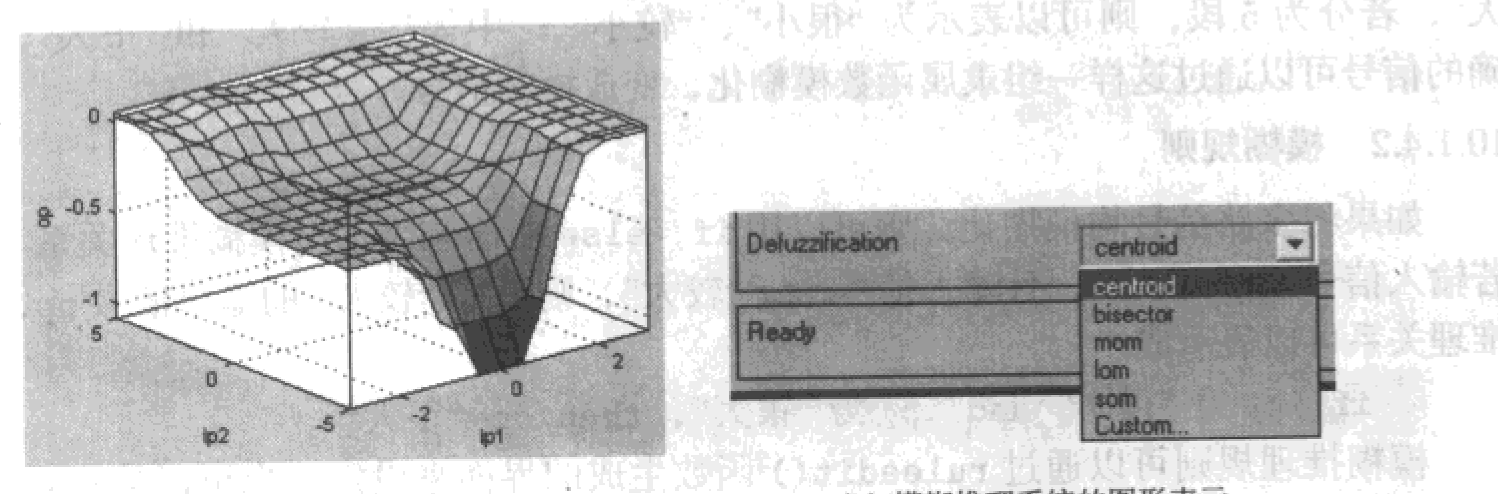


图 10-9　模糊规则编辑对话框



(a) 模糊规则的三维表面图　　　(b) 模糊推理系统的图形表示

图 10-10　模糊规则图形表示

[3, 2, 1, 1, 1]。

若用前面的规则生成一个规则矩阵 $\boldsymbol{R}$，则可以由下面的命令直接补加到模糊推理系统 fis 原有的规则后面。

```
fis=addrule(fis,R)
```

10.1.4.3　解模糊化

通过模糊推理可以得出模糊输出量 op，此模糊量可以通过指定的算法精确化，亦称解模糊化 (defuzzification)。模糊逻辑工具箱提供了多种解模糊化的算法，可以由如图 10-7 所示的对话框 Defuzzifications 栏目，亦即对话框中如图 10-10 (b) 所示的部分选择解模糊化算法。

按照上述的方式就可以建立起模糊推理系统的数据结构。可以由 File 菜单对模型进行处理，例如可以用 File → Export → To Disk 菜单项将其存成文件，后缀名为 fis。用户可以将前面编辑的模糊推理系统存储成 c10mfis.fis 文件。该工作还可以通过 `writefis()` 函数完成。还可以由 File → Export → To Workspace 菜单项将其存入 MATLAB 工作空间，存储时应该给出变量名。

模糊推理问题还可以用 MATLAB 函数 `evalfis()` 求解。该函数的调用格式为

```
y=evalfis(X,fis)
```

其中，$\boldsymbol{X}$ 为矩阵，其各列为各个输入信号的精确值，`evalfis()` 函数会对用户定义的模糊推理系统 `fis` 对这些输入信号进行模糊化，用该系统进行模糊推理，并将结果进行解模糊化，得出相应的精确输出信号 $\boldsymbol{y}$。

【例 10-8】 假设已经按上述方式建立起了模糊推理模型，在 x-y 平面内的 $(-3,-5)-(3,5)$ 区域内进行网格分割，试用此模糊推理系统绘制出输出的三维曲面。

【求解】 采用下面的语句可以先读入前面建立的模糊推理系统，并对感兴趣的 x-y 平面区域进行网格分割，将网格数据转换成列向量，再由 `evalfis()` 函数求出曲面的 z 坐标值，这样就可以用下面的语句绘制出三维曲面，如图 10-11 所示。

```
>> fff=readfis('c10mfis.fis');    % 读入模糊推理系统文件
   [x,y]=meshgrid(-3:.2:3,-5:.2:5);  % 进行网格分割
   x1=x(:); y1=y(:); z1=evalfis([x1 y1],fff);  % 模糊推理
   z=reshape(z1,size(x)); surf(x,y,z)   % 绘制曲面
```

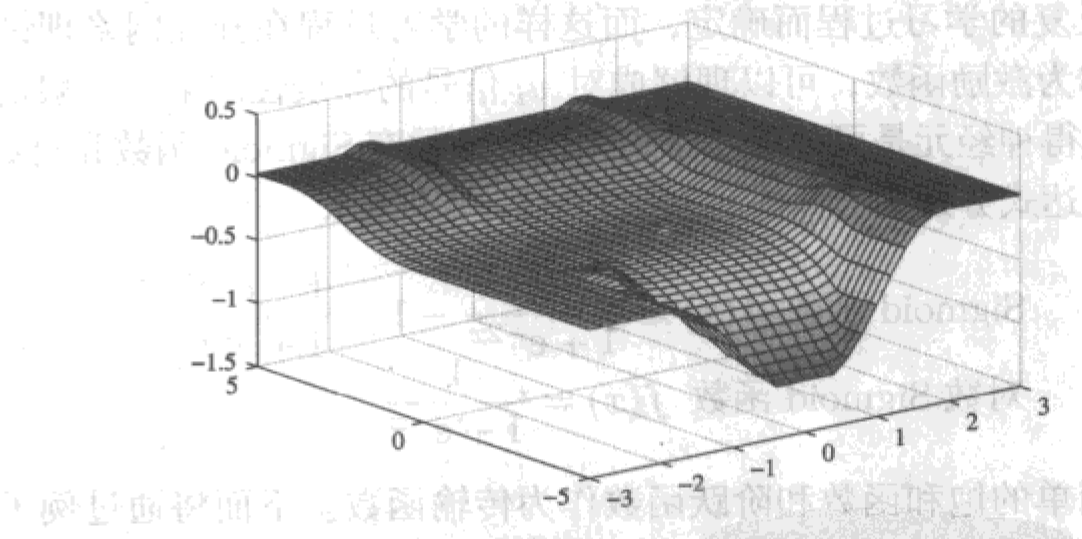


图 10-11 由模糊推理得出的输出曲面

10.2 神经网络及其在数据拟合中的应用

人工神经网络是在对复杂的生物神经网络研究和理解的基础上发展起来的。人脑是由大约 10^{11} 个高度互连的单元构成，这些单元称为神经元，每个神经元约有 10^4 个连接[12]。仿照生物的神经元，可以用数学方式表示神经元，引入人工神经元的概念，并由神经元的互连可以定义出不同种类的神经网络。限于当前的计算机水平，人工神经网络不可能有人脑那么复杂。

本节将首先介绍人工神经元和人工神经网络的数学结构，然后介绍神经网络的建立、训练与泛化的概念以及 MATLAB 语言的神经网络工具箱在解决这些问题中的应用，最后介绍一个用户界面来利用神经网络求解数据拟合中的问题。

10.2.1 神经网络基础知识

10.2.1.1 神经网络的概念及结构

单个人工神经元的数学表示形式如图 10-12 所示。其中，$x_1, x_2, \cdots, x_n$ 为一组输入信号，它们经过权值 w_i 加权后求和，再加上阈值 b，则得出 u_i 的值，可以认为该值为输入信号与阈值所构成的广义输入信号的线性组合。该信号经过传输函数 $f(\cdot)$ 可以得出神经元的输出信号 y。

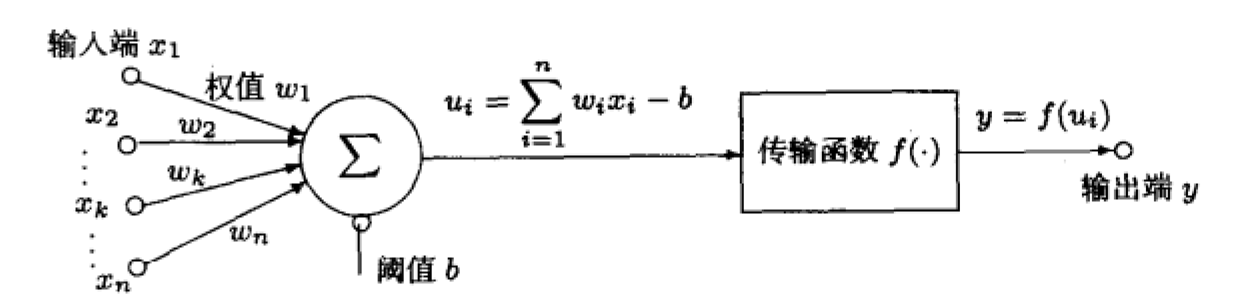


图 10-12 神经元的基本结构

在神经元中，权值和传输函数是两个关键的因素。权值的物理意义是输入信号的强度，若涉及多个神经元则可以理解成神经元之间的连接强度。神经元的权值 w_i 应该通过神经元对样本点反复的学习过程而确定，而这样的学习过程在神经网络理论中又称为训练。传输函数又称为激励函数，可以理解成对 u_i 信号的非线性映射，一般的传输函数应该为单值函数，使得神经元是可逆的。常用的传输函数有 Sigmoid 函数和对数 Sigmoid 函数，它们的数学表达式分别为

$$\text{Sigmoid 函数}\ f(x) = \frac{2}{1+\mathrm{e}^{-2x}} - 1 = \frac{1-\mathrm{e}^{-2x}}{1+\mathrm{e}^{-2x}}$$
$$\text{对数 Sigmoid 函数}\ f(x) = \frac{1}{1+\mathrm{e}^{-x}} \tag{10-2-1}$$

当然也可以使用简单的饱和函数和阶跃函数作为传输函数。下面将通过例子介绍各类传输函数的形状及基于 MATLAB 神经网络工具箱的绘制方法。

【例 10-9】试绘制出各种常用的传输函数曲线。

【求解】用下面的语句可以直接绘制出 Sigmoid 函数的曲线，如图 10-13 所示。

```
>> x=-2:0.01:2; y=tansig(x); plot(x,y)
```

用 `logsig()` 语句取代前面的 `tansig()` 函数则得出对数 Sigmoid 函数曲线。另外，由其他函数可以绘制出另外几种传输函数曲线，如图 10-13 所示，同时给出了绘制这些传输函数的 MATLAB 函数名。

由若干个神经元相互连接，则可以构成一种网络，称为神经网络。由于连接方式的不同，神经网络的类型也将不同。这里仅介绍前馈神经网络，因为其权值训练中采用误

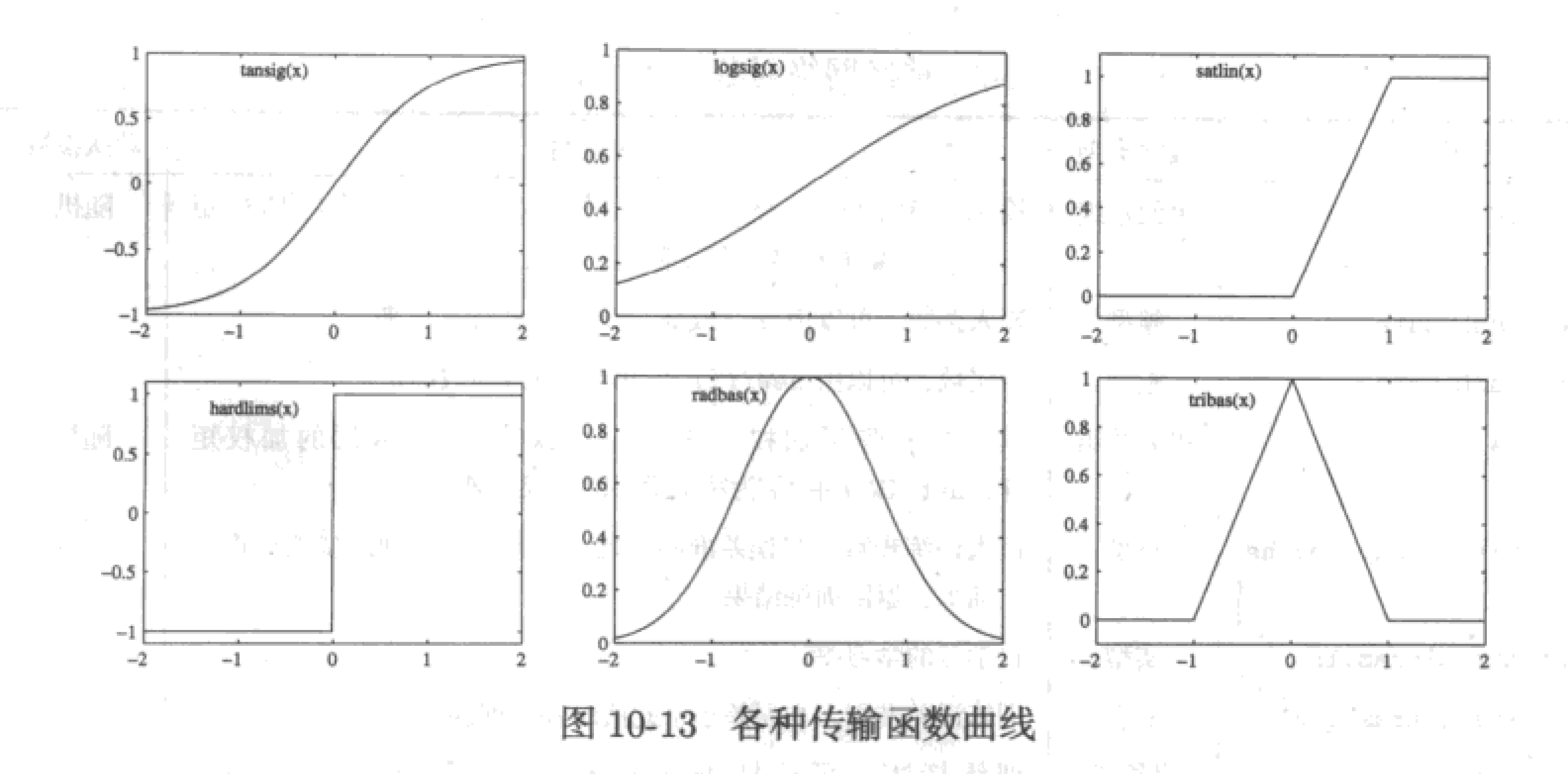


图 10-13 各种传输函数曲线

差逆向传播的方式，所以这类神经网络更多地称为反向传播 (back propagation) 神经网络，简称 BP 网。BP 网的基本网络结构如图 10-14 所示。这类的神经网络层数的定义和一般神经网络教材稍有差异，图 10-14 中给出的最后一个隐层实际上本身就是输出层，所以实际隐层数为 $k-1$。

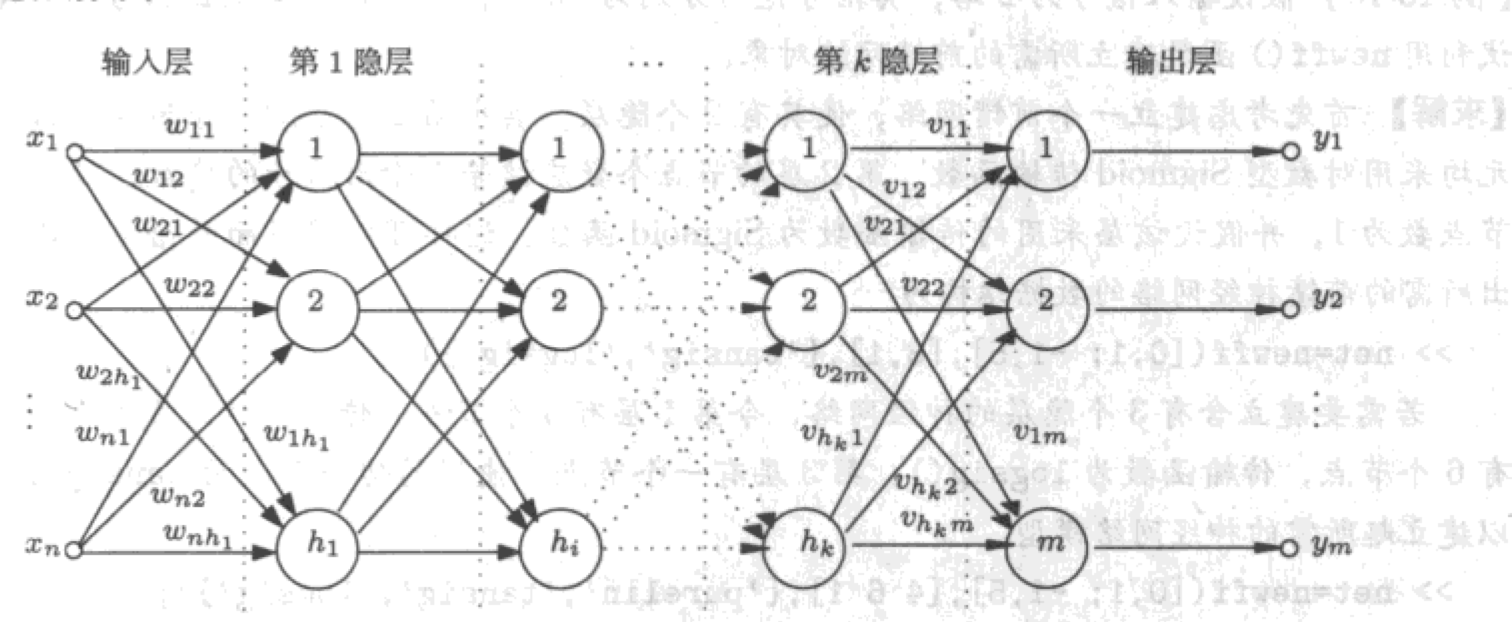


图 10-14 神经元的基本结构

利用 MATLAB 语言的神经网络工具箱提供的现成函数和神经网络类，建立起一个前馈的 BP 神经网络模型还是很容易的，可以使用 `newff()` 函数。具体的语句格式为

`net=newff(`$[\boldsymbol{x}_m,\boldsymbol{x}_M]$`,`$[h_1,h_2,\cdots,h_k]$`,`$\{f_1,f_2,\cdots,f_k\}$`)`

其中，$\boldsymbol{x}_m$ 和 $\boldsymbol{x}_M$ 分别为列向量，存储各个样本输入数据的最小值和最大值。若样本数据已知，还可以用 `min()` 和 `max()` 函数求出，第 2 个输入变量是一个行向量，用户可以将神经网络各层的节点数输入，单元的个数为隐层层数。函数的第 3 个输入变量为单元式数组，由若干个字符串构成，每个字符串对应于该层的传输函数类型。注意，这类要求同一层应该使用相同的传输函数。给了这些信息后，就可以构造出神经网络数据对象 `net`，该对象的一些重要属性在表 10-2 中给出。下面通过例子演示神经网络对象的建立。

表 10-2 神经网络对象的常用属性

属性名	数据类型	属性说明	默认参数
net.IW	单元数组	输入层和隐层加权，其中 net.IW{1} 为输入层的加权矩阵，net.IW{$i+1$} 为第 i 隐层的加权矩阵	随机
net.numInputs	整型	输入路数，可以由 x_m 或 x_M 维数自动计算出来	
net.numLayers	整型	隐层数，可以由 newff() 语句调用时的参数确定	
net.LW	单元数组	输入层和隐层加权，其中 net.IW{1} 为输入层的加权矩阵，net.IW{$i+1$} 为第 i 隐层的加权矩阵	随机
net.trainParam.epochs	整型	最大训练步数，当误差准则满足，即使未训练到此步骤也将停止训练，返回训练结果	100
net.trainParam.lr	实型	自学习的学习率	0.01
net.trainParam.goal	实型	训练误差准则，当误差小于此值时停止训练	0
net.trainFcn	字符串	训练算法，可选择 'traincgf'（共轭梯度法）、'train'（批处理训练算法）、'traingdm'（带动量的梯度下降算法）、'trainlm'（Levenberg-Marquardt 算法）等	'train'

【例 10-10】 假设输入信号为 2 路，其信号范围分别为 $[0,1]$ 和 $(-1,5)$，且输出信号为单路信号，试利用 newff() 函数建立所需的前馈网络对象。

【求解】 首先考虑建立一个前馈网络，使其有 2 个隐层，其中第 1 隐层有 8 个节点，且该层神经元均采用对数型 Sigmoid 传输函数，第 2 层的节点个数应该等于输出信号的路数，故只能选择其节点数为 1，并假设该层采用的传输函数为 Sigmoid 函数。这样可以用下面的语句可以立即建立出所需的前馈神经网络的数据结构为

```
>> net=newff([0,1; -1,5],[8,1],{'tansig','logsig'});
```

若需要建立含有 3 个隐层的神经网络，令第 1 层有 4 个节点，传输函数为线性函数，第 2 层有 6 个节点，传输函数为 logsig()，第 3 层有一个节点，使用的传输函数为 tansig()，这样可以建立起所需的神经网络模型。

```
>> net=newff([0,1; -1,5],[4 6 1],{'purelin','tansig','logsig'});
```

除了神经网络结构之外，还可以用下面的语句格式设定其他参数，如

```
net.trainParam.epochs=300, net.trainFcn='trainlm'
```

10.2.1.2 神经网络的训练与泛化

若建立了神经网络模型 net，则可以调用 train() 函数对神经网络参数进行训练。该函数的调用格式为

```
[net,tr,Y1,E]=train(net,X,Y)
```

其中，变量 $\boldsymbol{X}$ 为 $n\times M$ 矩阵，n 为输入变量的路数，M 为样本的组数，$\boldsymbol{Y}$ 为 $m\times M$ 矩阵，m 为输出变量的路数，$\boldsymbol{X},\boldsymbol{Y}$ 分别存储样本点的输入和输出数据。由样本点数据进行训练，则可以得出训练后的神经网络对象 net，且可以返回其他相关的内容，tr 为结构体数据，返回训练的相关跟踪信息，tr.epochs 为训练步数，tr.perf 为各步目标函数的

值。Y_1 和 E 矩阵分别返回由神经网络计算出的输出和误差矩阵。在训练过程中将每隔 25 步自动显示一次训练指标。训练结束后还可以用下面的语句绘制出目标值曲线：

```
plotperf(tr)
```

如果在给出的最大训练步数下无法得出满足要求的网络，则将给出错误的信息提示。用户可以再调用该函数一次，这时将以上次的训练结果加权矩阵为初值继续训练，用户可以循环调用该语句。如果误差在几次循环后仍无显著改善，则说明网络结构有问题，应该修改网络结构。

神经网络训练完成后，可以利用该网络对样本区域内的其他输入量求解其输出值，这种求值的方法称为神经网络的仿真或泛化 (generalization)，可以理解为利用神经网络进行数据拟合，对新的输入点数据 X_1 调用 sim() 函数进行泛化，得出这些输入点处的输出矩阵 Y_1，且

Y_1=sim(net,X_1)

神经网络是否成功不在于对样本点本身拟合误差的大小，而关键在于其泛化效果。如果对样本点以外的其他输入点均有较好的拟合，则说明该神经网络结构合理。否则，训练出来的神经网络没有应用价值。下面将通过例子来演示神经网络及其在数据拟合中的应用及神经网络控制参数对训练的影响。

【例 10-11】 考虑用例 8-25 中给出的数据，试用神经网络对其进行拟合。

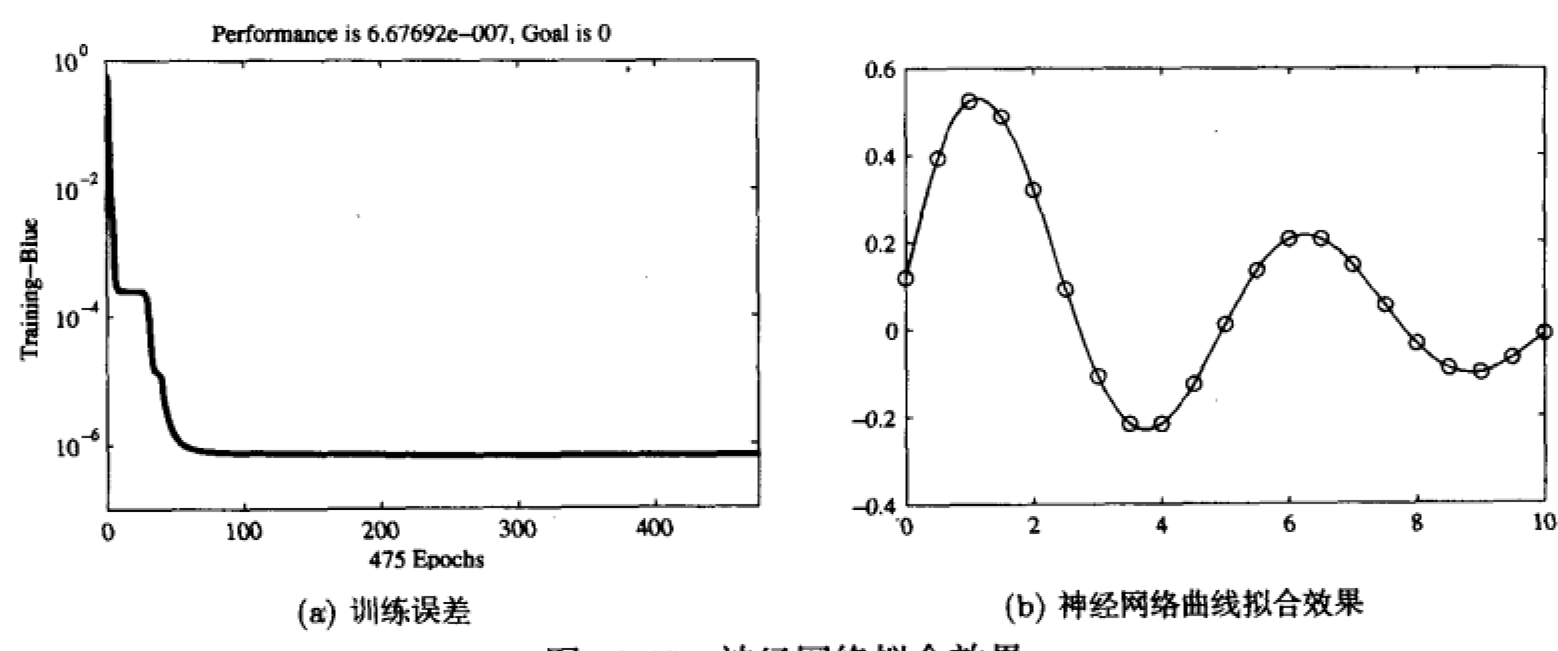
【求解】 可以用下面的语句输入样本点数据，并选择前馈神经网络。设有 2 个隐层，因为最后一个隐层实际上为输出层，所以其节点个数应该于输出路数一致，故节点数为 1。现在令第 1 隐层节点个数为 5，则可以用下面的语句进行神经网络训练，并选择更密集的输入数据进行泛化，得出如图 10-15 (a) 所示的训练误差，并得出如图 10-15 (b) 所示的泛化效果。可见，这样得出的拟合效果是令人满意的，和理论曲线之间看不出任何差异。

```
>> x=0:.5:10; y=0.12*exp(-0.213*x)+0.54*exp(-0.17*x).*sin(1.23*x);
   x0=[0:0.1:10]; y0=0.12*exp(-0.213*x0)+0.54*exp(-0.17*x0).*sin(1.23*x0);
   net=newff([0,10],[5,1],{'tansig','tansig'});
   net.trainParam.epochs=1000;   % 设置最大步数
   net=train(net,x,y);    % 训练神经网络
   figure; y1=sim(net,x0); plot(x,y,'o',x0,y0,x0,y1,':');
```

可以用下面的语句显示出神经网络的权值。

```
>> [net.IW{1}  net.LW{2,1}'] % 隐层权值和输出层权值
ans =
   0.70445907118894    -0.32029853336996
  -0.86985068184514    -0.17924013602994
   0.51921490556482     1.63158538435790
  -0.31618496850319     2.68718431457960
   0.99490706700810     1.11936254874762
```

选择不同的训练算法，则将得出不同的误差曲线，如图 10-16 (a) 所示。



(a) 训练误差

(b) 神经网络曲线拟合效果

图 10-15 神经网络拟合效果

```
>> net=newff([0,10],[5,1],{'tansig','tansig'}); net.trainParam.epochs=100;
   net.trainFcn='trainlm'; [net,b1]=train(net,x,y);
   net=newff([0,10],[5,1],{'tansig','tansig'}); net.trainParam.epochs=100;
   net.trainFcn='traincgf'; [net,b2]=train(net,x,y);
   net=newff([0,10],[5,1],{'tansig','tansig'}); net.trainParam.epochs=100;
   net.trainFcn='traingdx'; [net,b3]=train(net,x,y);
   semilogy(b1.epoch,b1.perf); hold on
   semilogy(b2.epoch,b2.perf,'--'); semilogy(b3.epoch,b3.perf,':')
```

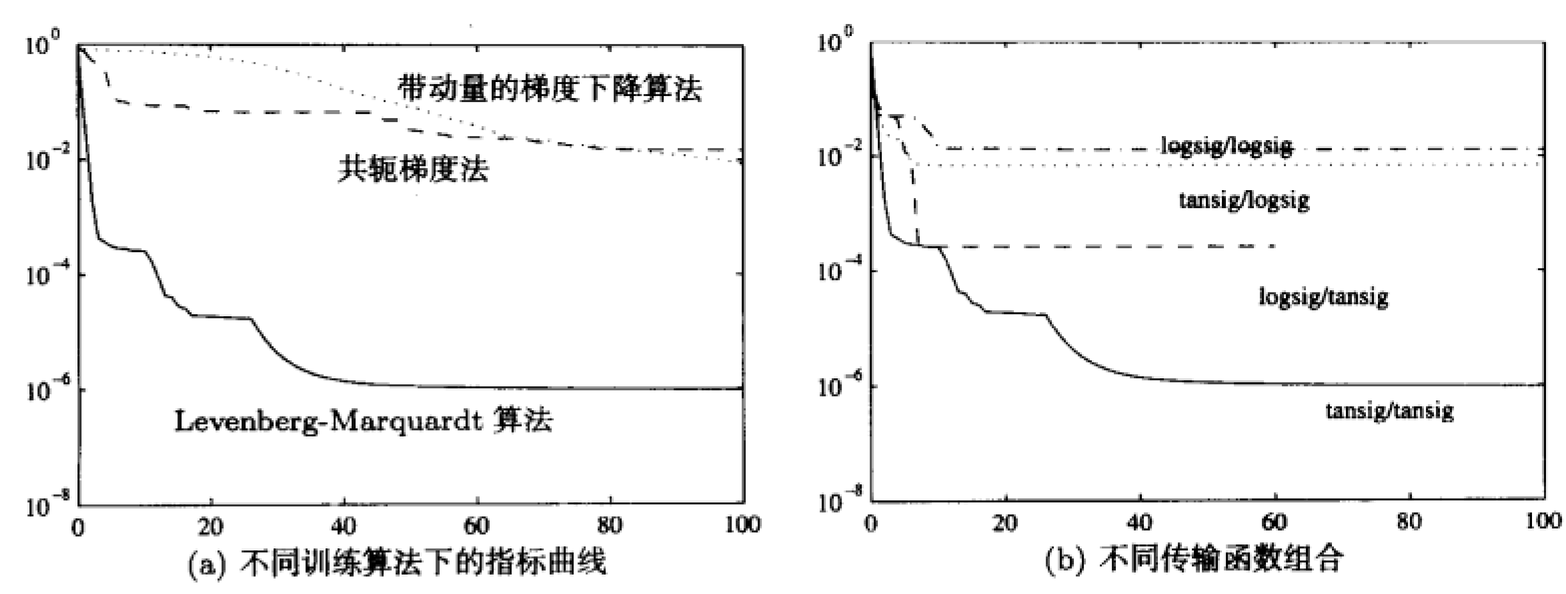
从训练效果看，用其他算法很多步数难以达到的训练效果用 Levenberg-Marquardt 算法可以较少步就能得出满意的效果。下面再讨论各层传输函数对训练的影响，可以给出如下命令对不同隐层组合进行试验，得出如图 10-16 (b) 所示的结果。可见，采用两层均采用 tansig() 函数的训练效果明显好于其他组合，故在实际训练中若没有特殊要求，应该采用 Sigmoid 函数。

```
>> net=newff([0,10],[5,1],{'tansig','logsig'}); net.trainParam.epochs=100;
   net.trainFcn='trainlm'; [net,b2]=train(net,x,y);
   net=newff([0,10],[5,1],{'logsig','tansig'}); [net,b3]=train(net,x,y);
   net=newff([0,10],[5,1],{'logsig','logsig'}); [net,b4]=train(net,x,y);
   semilogy(b1.epoch,b1.perf); hold on;  semilogy(b2.epoch,b2.perf,'--');
   semilogy(b3.epoch,b3.perf,':'); semilogy(b4.epoch,b4.perf,'-.')
```

若增加隐层节点个数，例如增加到 15，则可以重新建立神经网络，并进行训练，从训练结果看，只用 50 步就能得出极小的拟合误差，如图 10-17 (a) 所示。得出的曲线拟合结果如图 10-17 (b) 所示。

```
>> net=newff([0,10],[15,1],{'tansig','tansig'}); net.trainParam.epochs=100;
   net.trainFcn='trainlm'; [net,b2]=train(net,x,y);
   figure; y1=sim(net,x0); plot(x0,y0,x0,y1,x,y,'o')
```

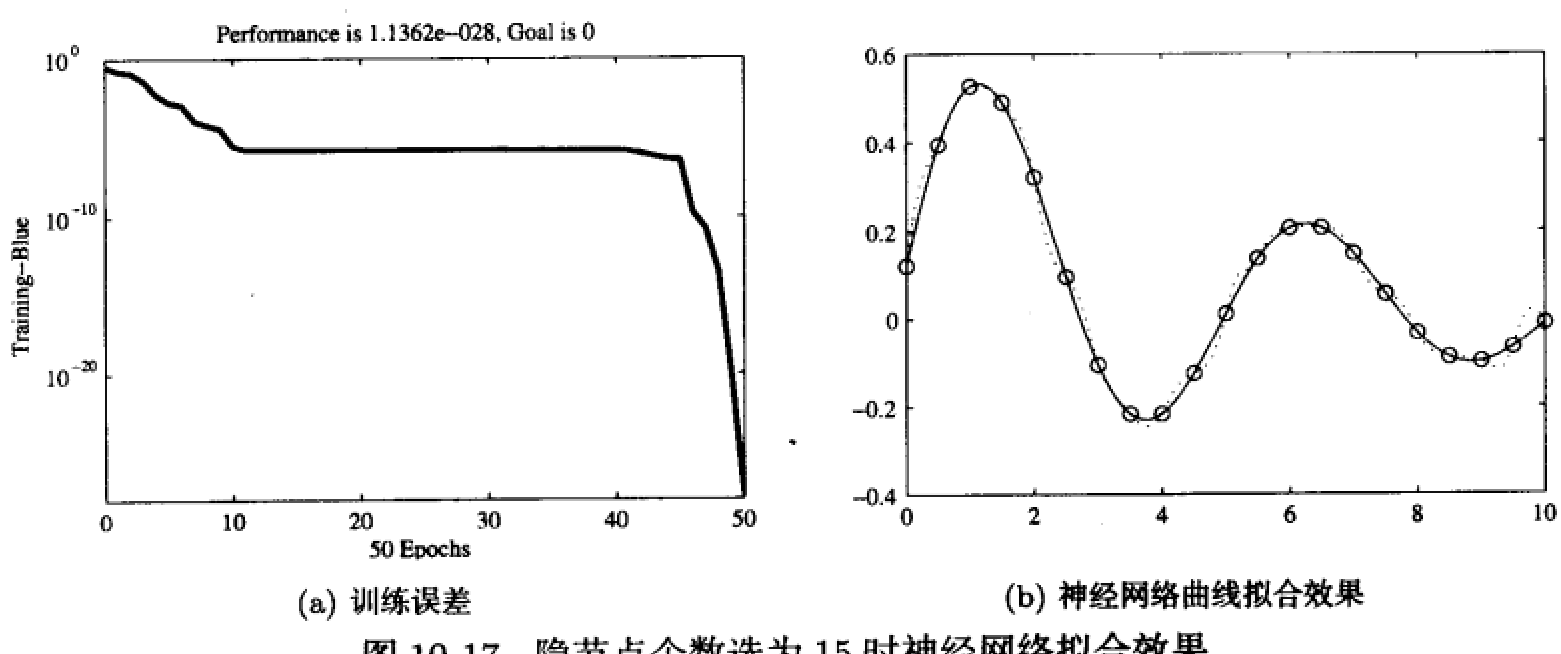
从得出的拟合结果看，似乎无限增大隐层节点个数就可以改进拟合效果。其实不然，增大节



(a) 不同训练算法下的指标曲线 (b) 不同传输函数组合

图 10-16 神经网络控制参数及其作用

点个数会改善对样本点的拟合，但对其他点的函数拟合将得出如图 10-17 (b) 所示的结果，亦即神经网络泛化出现了问题，故不能无限增加节点个数。不过节点个数如何选择至今没有公认的解析方法，只能根据实际情况用试凑方式选择。



(a) 训练误差 (b) 神经网络曲线拟合效果

图 10-17 隐节点个数选为 15 时神经网络拟合效果

【例 10-12】 试用神经网络对例 8-6 中给出的二元函数进行曲面拟合。

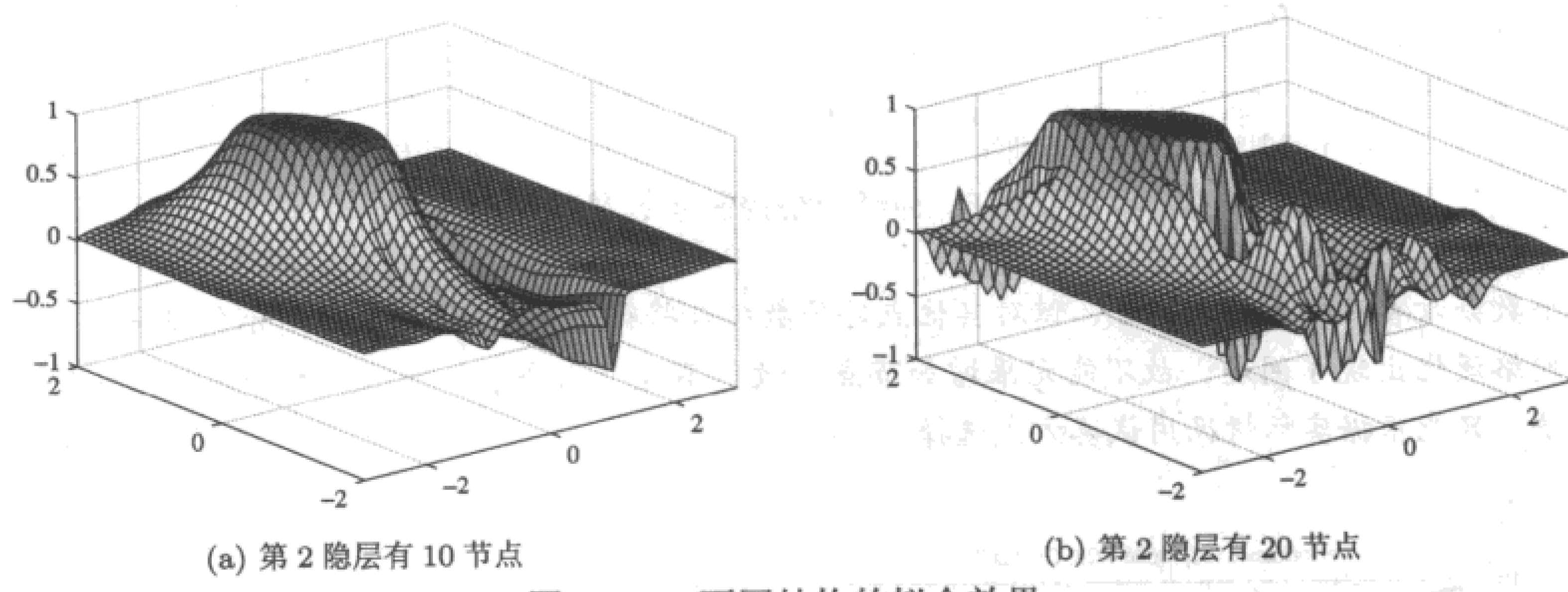
【求解】 先考虑用下面的语句输入样本数据，选择 3 隐层网络，第 1,2 层均有 10 个节点，采用 Sigmoid 函数作为传输函数，这样就可以用下面的语句对该网络进行训练，并得出如图 10-18 (a) 所示的泛化结果。

```
>> [x,y]=meshgrid(-3:.6:3, -2:.4:2); x=x(:)'; y=y(:)';
   z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y); % 注意这三个变量均应为行向量
   net=newff([-3 3; -2  2],[10,10,1],{'tansig','tansig','tansig'});
   net.trainParam.epochs=1000; net.trainFcn='trainlm';
   [net,b]=train(net,[x; y],z);    % 训练神经网络
   [x2,y2]=meshgrid(-3:.1:3, -2:.1:2); x1=x2(:)'; y1=y2(:)';
```

```
figure; z1=sim(net,[x1; y1]); z2=reshape(z1,size(x2)); surf(x2,y2,z2)
```

泛化效果不是很理想，部分点处有较大波动，现在设定第 2 层选择 20 个节点，则可以得出如图 10-18 (b) 所示的泛化效果。可见，泛化效果恶化，说明节点数选择过多。从总体拟合效果看，因为样本点不充足，不足以用神经网络得出较好的拟合结果，所以神经网络直接拟合效果较差。

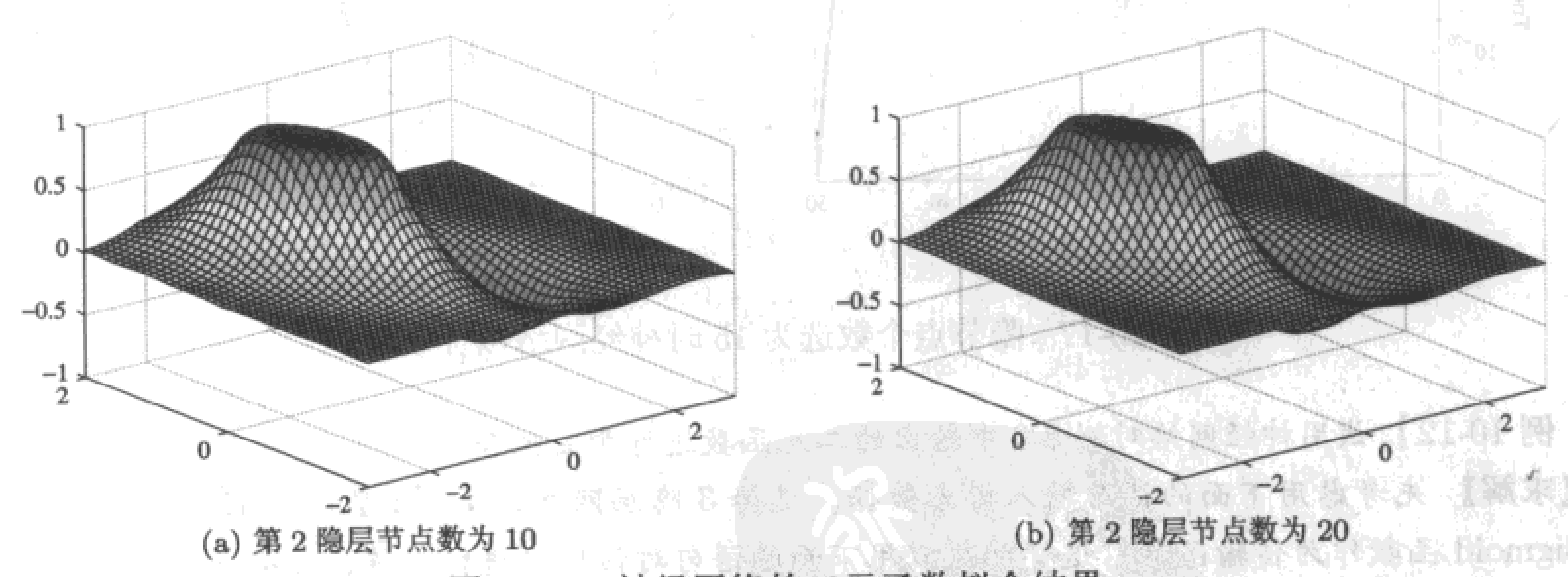
```
>> net=newff([-3 3; -2  2],[10,20,1],{'tansig','tansig','tansig'});
   [net,b]=train(net,[x; y],z);    % 训练神经网络
   z1=sim(net,[x1; y1]); z2=reshape(z1,size(x2)); surf(x2,y2,z2)
```



(a) 第 2 隐层有 10 节点　　(b) 第 2 隐层有 20 节点

图 10-18　不同结构的拟合效果

现在给出密集一点的样本点，则可以得出用下面的命令分别得出当第 2 层节点个数为 10 和 20 时，二元曲面的拟合效果，如图 10-19 (a)、图 10-19 (b) 所示。可见，拟合效果比前面的结果好，但远不如第 8 章中减少的样条插值的效果。



(a) 第 2 隐层节点数为 10　　(b) 第 2 隐层节点数为 20

图 10-19　神经网络的二元函数拟合结果

```
>> [x,y]=meshgrid(-3:.2:3, -2:.2:2); x=x(:)'; y=y(:)';
   z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
   net=newff([-3 3; -2  2],[10,10,1],{'tansig','tansig','tansig'});
   net.trainParam.epochs=100; net.trainFcn='trainlm';
   net=train(net,[x; y],z);
   [x1,y1]=meshgrid(-3:.1:3, -2:.1:2); a=x1;  x1=x2(:)'; y1=y2(:)';
```

```
z1=sim(net,[x1; y1]); z2=reshape(z1,size(a)); surf(x2,y2,z2)
net=newff([-3 3; -2  2],[10,20,1],{'tansig','tansig','tansig'});
net=train(net,[x; y],z);  % 修改节点个数后的泛化效果
figure; z1=sim(net,[x1; y1]); z2=reshape(z1,size(a)); surf(x2,y2,z2)
```

对这个例子来说，在当前测试的组合下效果远远差于例 8-6 中介绍的样条插值算法。用前馈神经网络拟合时，无论采用哪种训练算法，选择哪种网络和节点组合，得出的误差曲线基本均如图 10-20 所示，当误差准则等于 0.0021 时不再减少，所以不能得出令人满意的拟合效果。

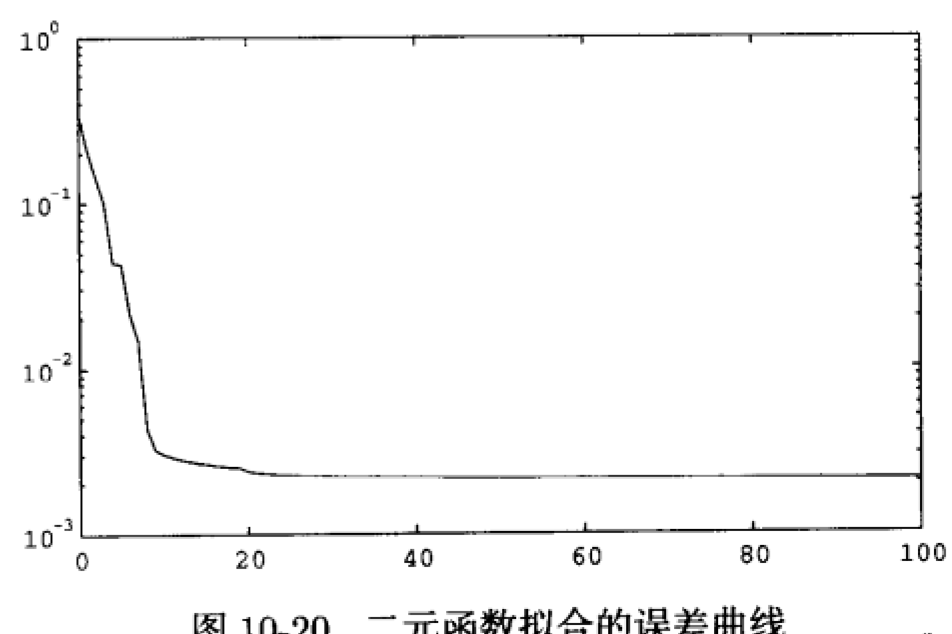


图 10-20 二元函数拟合的误差曲线

在神经网络研究中，经常有人引用这样的理论结果，即三层 BP 网络可以按任意精度逼近给定函数。不过从这样普通的例子看，即使选定 500 个隐层节点，试用了神经网络工具箱中提供的全部训练算法，也无法得出小于 10^{-3} 的误差，故而此例子作者没有得出任意精度的可行的神经网络。

10.2.2 神经网络界面

MATLAB 的神经网络工具箱提供了一个可以直接使用的程序。在 MATLAB 命令窗口中给出 nntool 命令，就可以打开一个如图 10-21 所示的图形用户界面，可以用该界面建立所需的神经网络模型，并可以由已知数据对该网络进行训练、仿真。下面将通过例子演示神经网络图形界面的使用。

【例 10-13】考虑用例 10-11 中给出的数据和神经网络拟合效果，试利用神经网络工具箱的 nntool 界面完成同样的拟合。

【求解】可以先输入已知数据，然后启动 nntool，将得出如图 10-21 所示的程序界面。可以通过这个界面对给定的数据进行神经网络拟合。

```
>> x=0:.5:10; y=0.12*exp(-0.213*x)+0.54*exp(-0.17*x).*sin(1.23*x);
   x0=[0:0.1:10]; y0=0.12*exp(-0.213*x0)+0.54*exp(-0.17*x0).*sin(1.23*x0);
   nntool     % 启动神经网络拟合界面
```

如果想利用神经网络拟合系统，首先需要将数据输入到此界面。这可以通过 Import 按钮来实现，单击该按钮将弹出如图 10-22 所示的对话框。将中间栏目下的 x, xx 两个现有的 MATLAB 工

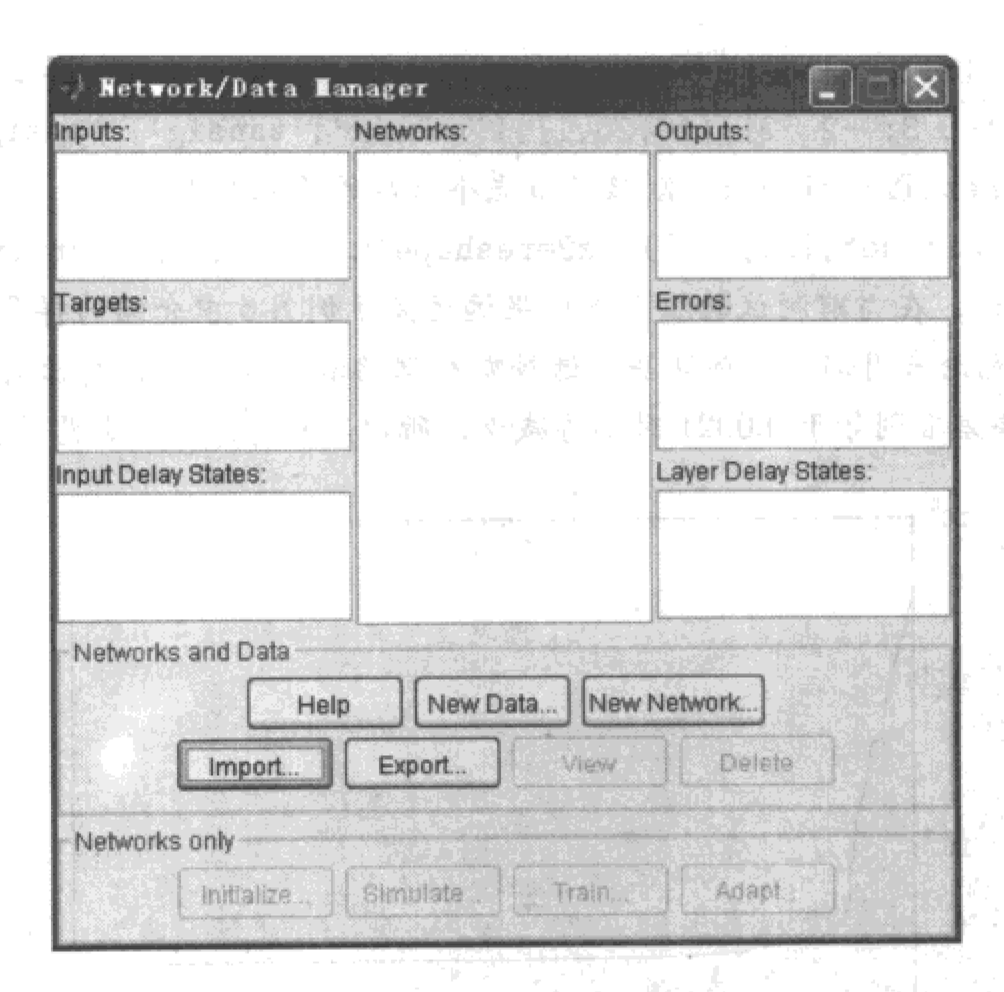


图 10-21　神经网络应用界面

作空间变量作为输入变量 (右面栏目的 Inputs) 输入，将变量 y 作为目标变量 (亦即 Targets) 输入到界面中。

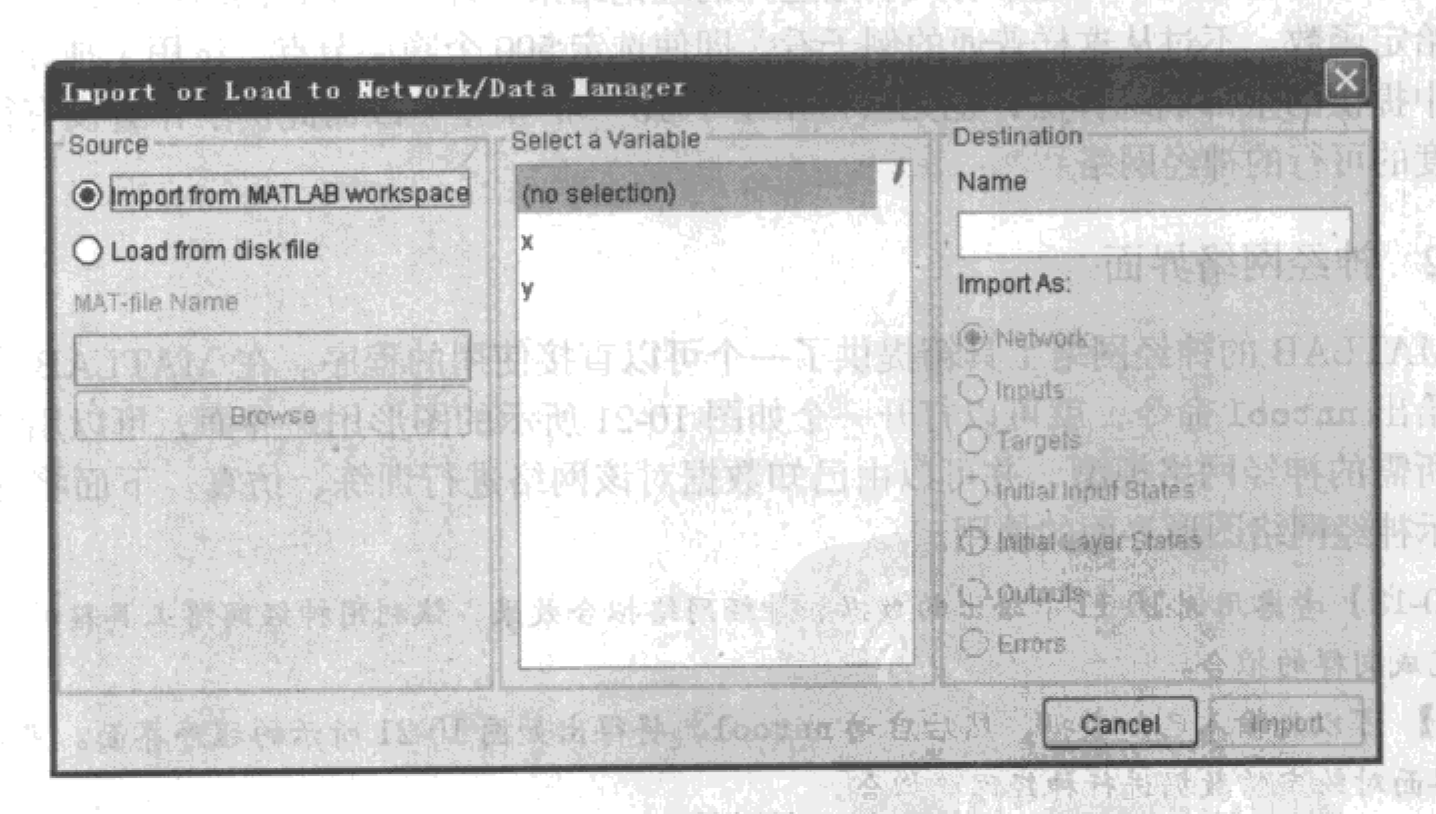
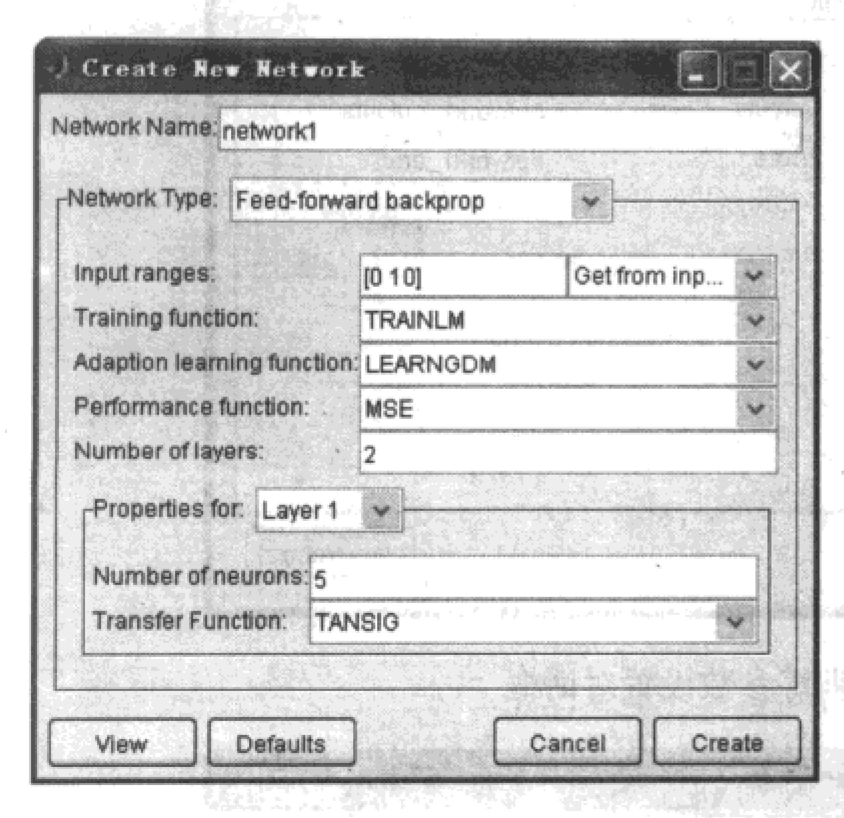


图 10-22　数据输入界面

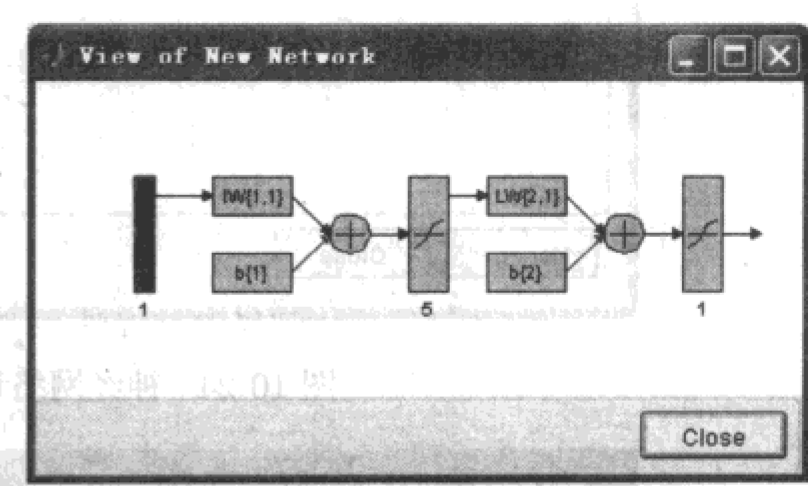
导入了所需的数据，则可以单击主界面中的 New Network 按钮来选择神经网络的结构，这样将得出如图 10-23 (a) 所示的界面。其中默认的网络结构是前馈型反向传播网络网络 (Feedforward Backprop)。保持各个默认的网络结构，将网络层数 (Number of layers) 设置成 2，在下面的列表框中分别选择不同的节点数，第 1 层选择 8 个节点数，第 2 层选择一个节点。在第 1 层中选择

Transfer Function 为 Logsig，第 2 层选择传递函数为 Pureline，单击 Create 按钮就可以确定神经网络结构，关闭此窗口。

神经网络结构确定后，单击 View 即可以显示神经网络结构，如图 10-23 (b) 所示。



(a) 神经网络结构设置对话框



(b) 神经网络结构

图 10-23 BP 网络结构设置与显示

现在可以训练神经网络了。单击 Train 按钮，将得出如图 10-24 所示的对话框。在对话框中需要输入训练用已知数据，如在 Inputs 和 Targets 栏目分别填写 x 和 y。另外，还可以单击 Training Parameters 标签指定神经网络训练的控制参数，得出如图 10-25 所示的对话框，例如将训练步数 epochs 设置为 1000，这时程序将自动训练网络参数，当误差指标满足时会自动停止训练，得出所需的参数。

单击 Train 按钮就可以开始训练，得出如图 10-15 (a) 所示的训练误差曲线。可见，这样的训练误差很小，将训练得出的网络模型输出 (Export) 到 MATLAB 环境中，这时将由下面的语句绘制出曲线拟合与泛化结果，如图 10-15 (b) 所示。

```
>> y1=sim(network1,x0); plot(x,y,'o',x0,y0,x0,y1,':')
```

10.3 遗传算法及其在最优化问题中的应用

遗传算法是基于进化论，在计算机上模拟生命进化机制而发展起来的一门新学科，它根据适者生存、优胜劣汰等自然进化规则搜索和计算问题的解[40]。该领域最早是由美国 Michigen 大学的 John Holland 于 1975 年提出的。遗传算法的基本思想是，从一个代表最优化问题解的一组初值开始进行搜索，这组解称为一个种群，种群由一定数量、通过基因编码的个体组成，其中每一个个体称为染色体，不同个体通过染色体的复制、交叉或变异又生成新的个体，依照适者生存的规则，个体也在一代一代进化，通过若干代的进化最终得出条件最优的个体。本节以一个较好的 MATLAB 工具箱 —— 遗传算法最优化工具箱为主要工具，先介绍遗传算法的基本概念，然后通过例子介绍其在求解最优

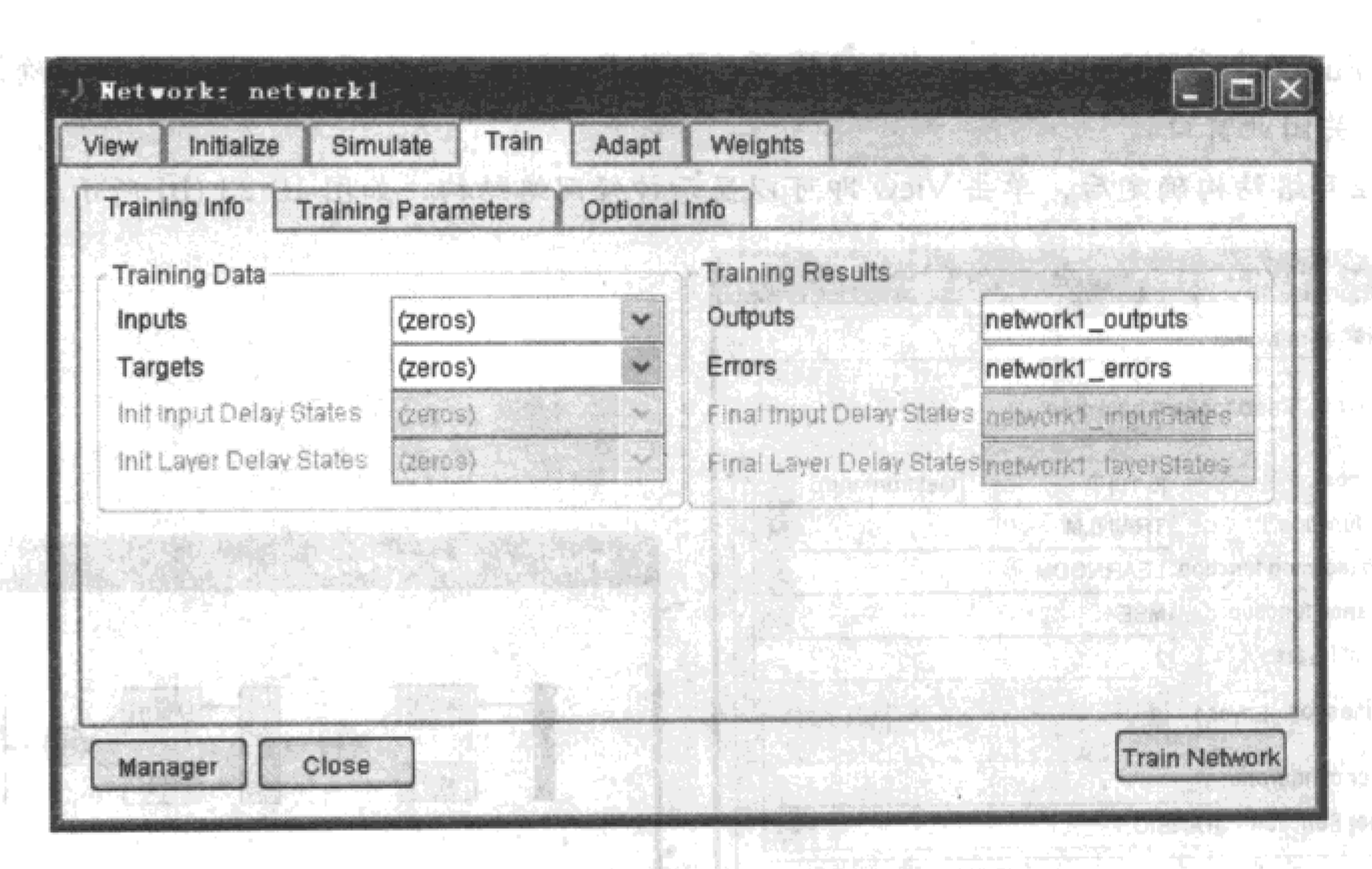


图 10-24 神经网络训练参数设置对话框

图 10-25 网络训练控制参数对话框

化问题中的应用。

10.3.1 遗传算法的基本概念介绍及 MATLAB 实现

新近推出的 MATLAB 7.0 版本带有遗传算法与直接搜索工具箱可以较好地解决与遗传算法相关的各类问题。但早期版本没有官方的工具箱，只有两个基于 MATLAB 语言的两个免费遗传算法工具箱。其一，英国 Sheffield 大学自动控制与系统工程系 Peter Fleming 教授与 Andrew Chipperfield 开发的遗传算法工具箱，实现了各种基本运算，规

范，说明书齐全，调用格式更类似于最优化工具箱中的函数；另一个，美国北 Carolina 州立大学 Christopher Houck, Jeffery Joines 和 Michael Kay 开发的遗传算法最优化工具箱，其函数 gaopt() 可以直接解决最优化问题①。对最优化问题来说，由于 GAOT 工具箱流传较广，已经能比较容易地解决最优化问题，所以这里还是建议采用该免费工具箱来解决基于遗传算法的最优化问题。

简单遗传算法的一般步骤为：

① 选择 N 个个体构成初始种群 $\boldsymbol{P}_0$，并求出种群内各个个体的函数值。染色体可以用二进制数组表示，也可以用实数数组来表示，种群可以由随机数生成函数建立。其实使用遗传算法求解函数 gaopt()，则会自动生成所需的初始种群 $\boldsymbol{P}_0$。

② 设置代数为 $i=1$，即设置其为第 1 代。

③ 计算选择函数的值，所谓选择即通过概率的形式从种群中选择若干个个体的方式。遗传算法最优化工具箱提供了 3 个选择函数，其中 roulette() 实现了轮盘选择算法，normGeomSelect() 函数实现了归一化几何选择方法，tournSelect() 实现了锦标赛形式的选择方式，normGeomSelect() 函数为默认选择函数。

④ 通过染色体个体基因的复制、交叉、变异等创造新的个体，构成新的种群 $\boldsymbol{P}_{i+1}$，其中复制、交叉和变异都有相应的 MATLAB 函数，gaopt() 函数选择其中默认的方法进行这样的处理，构成新的种群。

⑤ $i=i+1$，若终止条件不满足，则转移到步骤③继续进化处理。

和传统最优化算法比较，遗传算法主要有以下几点不同[5]：

① 不同于从一个点开始搜索最优解的传统的最优化算法。遗传算法从一个种群开始对问题的最优解进行并行搜索，所以更利于全局最优化解的搜索，但遗传算法需要指定各个自变量的范围，而不像最优化工具箱中可以使用无穷区间的概念。

② 遗传算法并不依赖于导数信息或其他辅助信息来进行最优解搜索，而只由目标函数和对应于目标函数的适应度水平来确定搜索的方向。

③ 遗传算法采用的是概率性规则而不是确定性规则，所以每次得出的结果不一定完全相同，有时甚至会有较大的差异。

本书研究的遗传算法使用的种群和个体表示均采用实值，这和经典遗传算法中采用的编码二进制值是不同的，所以无需对其进行编码和解码运算，且其求解问题的精度比二进制编码形式要高[16]。

10.3.2 遗传算法在求解最优化问题中的应用举例

这里将首先介绍遗传算法最优化工具箱中的 gaopt() 函数在求解最优化问题中的应用。之所以使用该函数是因为该函数调用简单。即使对遗传算法理解不多，甚至不知道染色体如何选择，如何进行交叉和变异，如何进行选择等关于遗传算法的最基本知识，但利用 MATLAB 语言描述出目标函数，就可以得出最优解。和最优化工具

①该工具箱的主函数名为 ga()，但该函数名与 MATLAB 7.0 中遗传算法和直接搜索工具箱中的函数同名，故这里将其改名为 gaopt()。

箱不同，gaopt() 函数能求解的是问题是最大化问题，所以在编写目标函数时应该注意。gaopt() 函数最基本的调用格式为

```
[a,b,c]=gaopt(bound,fun)
```

其中，bound=[x_m,x_M] 为求解区间下界 x_m 和上界 x_M 构成的矩阵，fun 为字符串，表示用户编写的目标函数，其写法与最优化工具箱的目标函数写法相近，但结构不完全相同，后面将通过例子来描述之。返回的 a 为搜索的结果向量，由搜索出的最优 x 向量与目标函数构成，b 为搜索的最终种群，c 为搜索中间过程参数表，其第一列为代数，后面各列分别为该代最好的个体与目标函数的值，可以认为是寻优的中间结果。当然该函数还有其他的调用格式，例如需要用户自己声明遗传算法的编码、选择等信息，设定初始种群、交叉率、变异率，还可以选择最大的代数。

MATLAB 7.0 的遗传算法与直接搜索工具箱也提供了 ga() 函数，其调用格式与最优化工具箱中的相应函数类似，可以同样利用遗传算法处理最优化问题。该函数的调用格式为

```
[x,f,flag,out]=ga(fun,n,opts)
```

其中，fun 为描述目标函数的 MATLAB 函数，其格式与最优化工具箱一致，n 为自变量个数，opts 为遗传算法控制选项，可以调用 gaoptimset() 函数设置各种选项。例如，用其 Generations 属性可以设定最大允许的代数，InitialPopulation 属性可以设置初始种群，用 PopulationSize 属性可以给定种群的规模，用 SelectionFcn 属性可以定义选择函数等等。函数调用结束后，返回的 x 为搜索的结果，若返回的 flag 大于 0，则表示求解成功，否则求解出现问题。

遗传算法与直接搜索工具箱还提供了 gatool() 函数，该函数将打开如图 10-26 所示的遗传算法程序界面。可以用可视的方式设置遗传算法的各类参数，如初始种群、交叉变异、选择函数等可以通过列表框与编辑框直接填写。单击 Start 按钮就可以开始开始遗传算法搜索，得出最优结果。

【例 10-14】考虑一个简单的一元函数最优化问题求解，$f(x)=x\sin(10\pi x)+2,\ x\in(-1,2)$，试求出 $f(x)$ 取最大值时 x 的值。

【求解】用下面的语句可以绘制出求解区间内的目标函数的曲线，如图 10-27 所示。可以看出，该曲线为振荡曲线，存在很多极值点。

```
>> ezplot('x*sin(10*pi*x)+2',[-1,2])
```

因为最优化工具箱的搜索函数需要给出初值，所以对不同的初值可能得出不同的搜索结果，例如可以给出如下的语句试测不同初值，得出的结果如表 10-3 所示。

```
>> f=inline('-x.*sin(10*pi*x)-2','x'); v=[];
   for x0=[-1:0.8:1.5,1.5:0.1:2]
      x1=fmincon(f,x0,[],[],[],[],-1,2); v=[v; x0,x1,f(x1)];
   end
```

可见，随意选择一个初值很难得出全局最优解，故用传统的寻优方式不一定能得出满意的结果。

利用遗传算法函数 gaopt()，完全选择默认选项，则可以编写一个文件描述目标函数如下：

Genetic Algorithm Tool

File Help

Fitness function: @c10mga3a

Number of variables: 4

Plots

Plot interval: 1

Best fitness　Best individual　Distance

Expectation　Genealogy　Range

Score diversity　Scores　Selection

Stopping

Custom function:

Run solver

Use random states from previous run

Start　Pause　Stop

Current generation: 5500

Status and results:

GA running.
GA terminated.
Fitness function value: 2.916430447810128E-12
Optimization terminated: maximum number of generati

Final point:

1	2	3	4
0.00012	-0.00012	0.00057	0.00057

Export to Workspace...

Options:

Population

Population type: Double Vector

Population size: 80

Creation function: Uniform

Initial population: []

Initial scores: []

Initial range: [0; 1]

Fitness scaling

Selection

Reproduction

Mutation

Crossover

Migration

Hybrid function

Stopping criteria

Output function

Display to command window

Vectorize

Quick reference:

Fitness function is the objective function you want to minimize. You can specify the function as a function handle of the form @objfun, where objfun.m is an M-file that returns a scalar.

Number of variables is the number of independent variables for the fitness function.

Plot Functions

Plot functions enable you to plot various aspects of the genetic algorithm as it is executing. Each one will draw in a separate axis on the display window. Use the **Stop** button on the window to interrupt a running process.

- **Plot interval** specifies the number of generations between successive updates of the plot.
- **Best fitness** plots the best function value in each generation versus iteration number.
- **Expectation** plots the expected number of children versus the raw scores at each generation.
- **Score diversity** plots a histogram of the scores at each generation.
- **Stopping** plots stopping criteria levels.
- **Best individual** plots the vector entries of the individual

图 10-26 遗传算法工具箱的设计界面

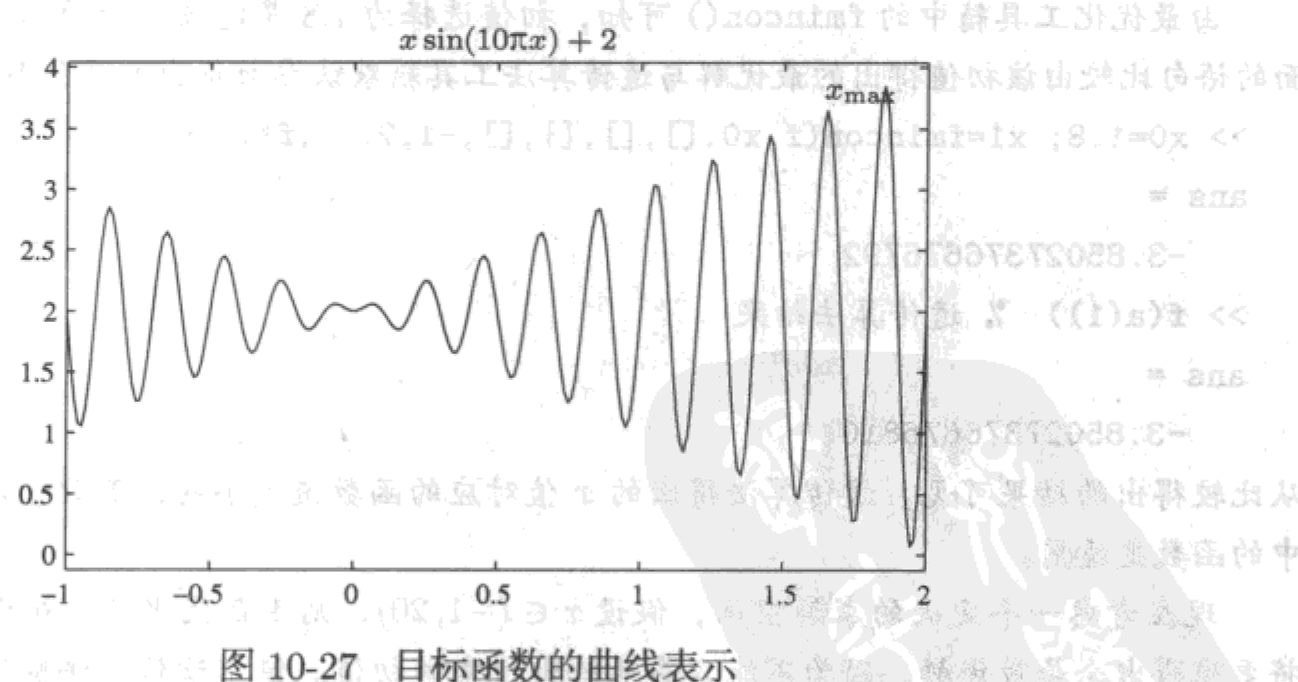


图 10-27 目标函数的曲线表示

```
function [sol,y]=c10mga1(sol,options)
x=sol; y=x.*sin(10*pi*x)+2;
```

这样，完全用默认参数调用遗传算法工具箱中的 gaopt() 函数，而不对其编码等做任何指

表 10-3 不同初值 x_0 下搜索到的最优解及目标函数值

x_0	搜索解 x_1	目标函数 $f(x_1)$	x_0	搜索解 x_1	目标函数 $f(x_1)$	x_0	搜索解 x_1	目标函数 $f(x_1)$
-1	-1	-2	1.4	1.450698306	-3.450349229	1.7	1.250809692	-3.250405044
-0.2	-0.6515538221	-2.650777689	1.5	0.2539684622	-2.251997259	1.8	1.850547452	-3.850273767
0.6	0.6515537909	-2.650777689	1.6	1.65061375	-3.650306929	1.9	0.4522326498	-2.451120675

定，则可以得出如下的结果[①]：

```
>> [a,b,c,d]=gaopt([-1,2],'c10mga1'); a,c
a =
    1.85054746647533    3.85027376676810
```

其 c 参数单独显式的可读性不佳，可以将其表示成表 10-4 的形式。可见，对此例来说，从第 13 代开始就可以得出较精确的结果，通过 100 代的搜索，可以得出相当高精度的寻优结果。

表 10-4 遗传算法搜索中间结果

代	x	$f(x)$	代	x	$f(x)$	代	x	$f(x)$
1	1.866429742	3.623276856	8	1.855582779	3.827116063	19	1.850630071	3.850267532
2	1.857329778	3.808304445	9	1.855325444	3.829420164	21	1.850569848	3.850273309
3	1.856754984	3.815102341	10	1.846815256	3.837579341	···	···	···
5	1.856200451	3.821095566	11	1.852559605	3.84657337	98	1.850547236	3.850273767
7	1.855683776	3.826178928	13	1.850438152	3.85026285	100	1.850547466	3.850273767

由最优化工具箱中的 fmincon() 可知，初值选择为 1.8 是能较精确的结果。这样，可以由下面的语句比较由该初值得出的最优解与遗传算法工具箱默认参数得出的最优解。

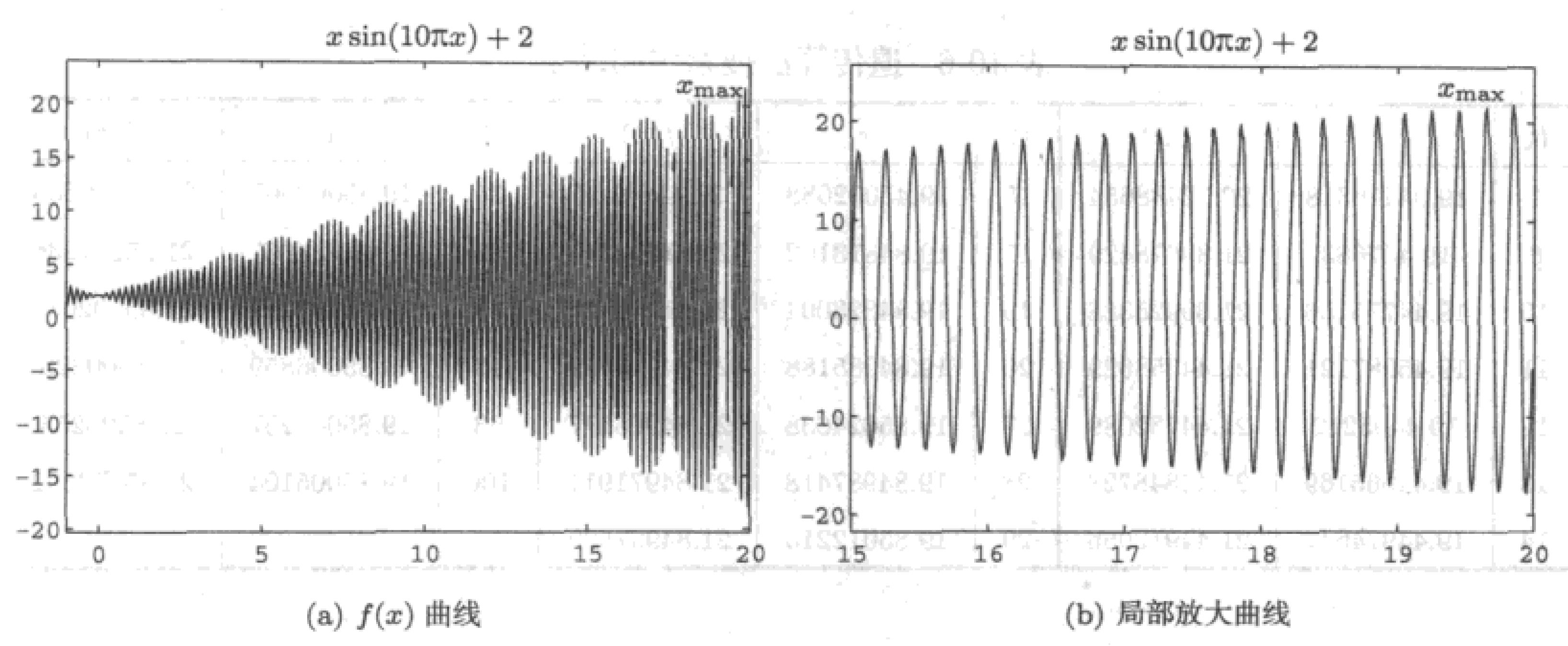
```
>> x0=1.8; x1=fmincon(f,x0,[],[],[],[],-1,2,'',ff); f(x1)
ans =
   -3.85027376676792
>> f(a(1))  % 遗传算法结果
ans =
   -3.85027376676810
```

从比较得出的结果可见，遗传算法得出的 x 值对应的函数值更小些，因此可以认为该方法对本例中的函数更适用。

现在考虑一个更大的求解空间，假设 $x \in (-1,20)$，则用最优化工具箱中的 fmincon() 函数将更难得出全局最优解，因为不能容易地给出搜索的初值。但用遗传算法则没有这样的问题，只需给出如下语句即可。

```
>> [a,b,c,d]=gaopt([-1,20],'c10mga1'); a,c
```

① 前面已经指出，遗传算法中有随机性因素，所以调用该算法每次得出的结果均不相同，但大的趋势应该是一样的。阅读本部分时不要指望您得出的结果和书上的完全一致。



(a) $f(x)$ 曲线　　(b) 局部放大曲线

图 10-28 变换区间后的目标函数曲线

```
a =
    19.45005207932562  21.45002604651104
```

其 c 参数可以表示成表 10-5 的形式。

表 10-5 遗传算法搜索中间结果

代	x	$f(x)$	代	x	$f(x)$	代	x	$f(x)$
1	18.04650679	19.93794526	13	18.8529686	20.77103967	29	19.45008417	21.45001617
3	18.05080536	20.04502802	17	19.05252045	20.99282365	40	19.45004021	21.45002469
6	18.05063177	20.04707655	20	19.0522703	21.00383083	44	19.45005193	21.45002605
11	18.45004062	20.4500256	22	19.45056616	21.44748956	100	19.45005208	21.45002605

其实，将该曲线局部放大，将得出如图 10-28 (b) 所示的表示形式。可见，这个区间内使得函数取最大值的 x 值应该大于 19.5，所以这样用遗传算法得出的并非全局最大值，而为次最优值。由此可见，遗传算法也不能保证获取全局最优值。

现在改变遗传算法的求解区间，则可以得出如下命令：

```
>> [a,b,c,d]=gaopt([12,20],'c10mga1'); a,c
a =
   19.85005104270946  21.85002552164857
```

【例 10-15】 试求函数 $f(x)=(x_1+x_2)^2+5(x_3-x_4)^2+(x_2-2x_3)^4+10(x_1-x_4)^4$ 的最小值。

【求解】 可以按照遗传算法工具箱编写如下函数，描述最优化问题的目标函数。注意，这里应该描述成目标函数的最大值。

```
function [sol,f]=c10mga3(sol,options)
x=sol(1:4);
f=-(x(1)+x(2))^2-5*(x(3)-x(4))^2-(x(2)-2*x(3))^4-10*(x(1)-x(4))^4;
```

使用遗传算法函数 gaopt()，并设定自变量的求解范围为 $-1\leqslant x_i\leqslant 1$，$i=1,2,3,4$，则可以由下

表 10-6 遗传算法搜索中间结果

代	x	$f(x)$	代	x	$f(x)$	代	x	$f(x)$
1	19.04416318	20.72488654	15	19.45002088	21.4500167	35	19.85009565	21.85000603
6	19.447683	21.39618429	17	19.84813102	21.81392715	37	19.8500877	21.85001236
10	19.44775156	21.39925388	19	19.84823094	21.81758539	43	19.85007289	21.85002085
11	19.45087121	21.44358629	20	19.84985188	21.84963699	56	19.85006859	21.8500225
12	19.4495212	21.44732089	27	19.85024638	21.84965177	63	19.85005257	21.8500255
13	19.44965169	21.44848728	28	19.84987418	21.84971911	100	19.85005104	21.85002552
14	19.44974647	21.44912956	29	19.85012215	21.84997599			

面的函数求解最优化问题，得出如下的结果，且部分中间结果如表 10-7 所示。事实上，该函数的精确解为 $x_1 = x_2 = x_3 = x_4 = 0$，但用遗传算法不能搜索到该点，只能得出次最优值。

```
>> [a,b,c,d]=gaopt([-1,1; -1 1; -1 1; -1 1],'c10mga3'); a,c
a =
   0.05511774654 -0.05428278824 0.01486592451 0.01508706783 -0.00007644215
```

表 10-7 遗传算法搜索部分中间结果

代	x_1	x_2	x_3	x_4	$f(x_1,x_2,x_3,x_4)$
1	-0.58664839284	0.60535582738	0.028407550371	-0.073658816991	-0.83549958111
4	-0.01182138153	-0.048357871289	0.21168424008	0.30791546876	-0.20395460867
9	-0.02498777978	-0.033384575447	0.11281202834	0.10206772219	-0.011090989881
13	-0.02498777978	0.0042179390678	0.11281202834	0.10206772219	-0.0060176098015
18	-0.0083633004824	0.013245084818	0.02411260887	0.023000817182	-4.118625464910^{-5}
24	-0.0082541665169	0.0050231349179	0.02411260887	0.023000817182	-2.964632067910^{-5}
28	-0.0082541665169	0.008935952083	0.02411260887	0.023000817182	-1.857090777210^{-5}
37	-0.00067214193564	0.0015648793181	0.02398676301	0.024507332569	-1.081027149610^{-5}
40	-0.002743469815	0.0034270462778	0.024021142751	0.024095768707	-9.646255329910^{-6}
83	-0.00079248828595	0.0026278066001	0.021397761566	0.02118842351	-8.525152175210^{-6}
85	-0.0015381099099	0.0027951007597	0.021397761566	0.02118842351	-7.02692040410^{-6}
89	-0.0019644292217	0.0015052840869	0.019625896238	0.019461658039	-4.483255974310^{-6}
100	-0.0017892249183	0.0019037994674	0.019465754048	0.019467860996	-3.934738981710^{-6}

上述的求解结果显然误差很大，其最主要的原因是最大允许的代数 (默认值为 100) 太小。另外，求解的区间太小，使得得出解的可信度降低。也就是说，允许的求解区域是 $[-1,1]$，如果选择更大的求解区域将得出更不精确的结果。

现在考虑 gaopt() 函数稍复杂一点的调用格式，如下：

$$[\boldsymbol{x},\mathrm{b},\mathrm{c},\mathrm{d}]=\mathrm{gaopt}(\mathrm{bound},\mathrm{fun},\boldsymbol{p},\boldsymbol{v},\boldsymbol{P}_0,\mathrm{fun1},\boldsymbol{n})$$

其中，p 可以给目标函数增加附加参数，这些参数必须和目标函数对应，v 为精度及显示控制向量，P_0 为初始种群，fun1 为终止函数的名称，默认值为 'maxGenTerm'，n 为最大的允许代数。当然还有其他的调用参数的用户设置格式，如选择函数及参数、变异函数及参数等，在这里不具体介绍，读者可以参见文献 [16]。

【例 10-16】 仍考虑例 10-15 中给出的最优化问题，试设置更多的允许代数，观察寻优的结果，并和最优化搜索算法得出结果在时间、精度上的差异。

【求解】 考虑新的求解区域，$-1000 \leqslant x_i \leqslant 1000$，由于求解范围增大，采用默认的代数 100 会有问题，所以可以考虑用新介绍的求解格式。显然，目标函数由 $\boldsymbol{x}$ 就能惟一确定，而无需附加参数，故令 $\boldsymbol{p}$=[] 即可。同样，因为不想改变初始种群，控制向量等，可以将它们设置成空向量。这样，若想指定 2000 代为最大进化代数，则可以给出如下的命令，得出的中间结果如表 10-8 所示。

```
>> tic, xmM=[-ones(4,1),ones(4,1)]*1000;
   [a,b,c,d]=gaopt(xmM,'c10mga3',[],[],[],'maxGenTerm',2000);
   a(1:4), dd=[c(1:100:end,:); c(end,:)], toc
a =
  -0.00093788501077   0.00084347409788  -0.01565338248119  -0.01569190233777
Elapsed time is 29.883000 seconds.
```

表 10-8 遗传算法搜索部分中间结果

代	x_1	x_2	x_3	x_4	$f(x_1,x_2,x_3,x_4)$
1	299.261366622	-139.695834899	-75.5914398821	272.888321091	-5487787.82563
211	259.334708142	-133.958552709	-63.9282289435	255.140011267	-529224.38675
418	207.950553013	-107.100526442	-50.7673718192	204.4965359	-338351.978321
633	183.944398697	121.408588645	63.0882570778	183.712206401	-166507.956154
828	30.5439763104	62.3298177457	31.7096211527	32.1807061178	-8699.82405036
1012	25.9003394774	41.1500210753	21.4199283474	24.9266965028	-4574.37877164
1211	19.218615649	26.8881841847	14.8795410924	20.0214636561	-2330.11994456
1390	6.33506176436	8.75276639595	5.30792816069	5.98777957032	-242.14756723
1559	0.317019478936	0.133356773142	0.42062335921	0.624193867527	-0.750183675915
1825	0.00172291111621	-0.000400676613735	-0.0233512636645	-0.0234295772108	-1.03776×10^{-5}
2000	-0.000937885010771	0.000843474097882	-0.0156533824812	-0.0156919023378	-1.55859×10^{-6}

可见，在 1390 代左右的误差都一直很大，所以用默认的 100 代处理不了这样大求解区间的问题，必须增大代数，然而若达到较高求解精度，则再进一步增加最大允许的代数也不会显著改善求解的精度了。

利用 MATLAB 7.0 提供的遗传算法求解函数 ga()，则可以编写出如下的目标函数

```
function f=c10mga3a(x)
f=(x(1)+x(2))^2+5*(x(3)-x(4))^2+(x(2)-2*x(3))^4+10*(x(1)-x(4))^4;
```

这样，调用 ga() 函数可以得出如下的结果

```
>> x=ga(@c10mga3a,4)
Optimization terminated: maximum number of generations exceeded.
x =
   0.00957762093689  -0.01670543425362   0.01026555932851   0.01072888292002
```

还可以人为指定一些搜索参数，如进化代数选择为 2000，种群选择为 80 个体，而交叉函数选择为启发式算法等，这样由下面的语句可以得出更精确的结果。

```
>> ff=gaoptimset; ff.Generations=2000; ff.PopulationSize=80;
ff.CrossoverFcn=@crossoverheuristic; x=ga(@c10mga3a,4,ff)
x =
   1.0e-003 *
  -0.22578529109784   0.22571071964788  -0.94344267150266  -0.94349063059752
Elapsed time is 7.721000 seconds.
```

现在考虑用第 6 章的无约束最优化求解功能，可以由如下的命令直接求出原问题的解，其精度明显高于遗传算法的结果，且求解时间大大减少。

```
>> f=inline...  % 目标函数描述
    ('(x(1)+x(2))^2+5*(x(3)-x(4))^2+(x(2)-2*x(3))^4+10*(x(1)-x(4))^4','x')
   ff=optimset; ff.MaxIter=10000; ff.TolX=1e-7;
   tic, x=fminsearch(f,10*ones(4,1),ff); toc; x'
Elapsed time is 0.751000 seconds.
ans =
  1.0e-006 *
    0.03039572499758  -0.03039585246164  -0.75343487601326  -0.75343518285272
```

【例 10-17】 试求解下面的非线性最优化问题。

$$\min_{(x,y)\ \text{s.t.}\left\{\begin{array}{l}-1\leqslant x\leqslant 3\\ -3\leqslant y\leqslant 3\end{array}\right.} \sin(3xy)+xy+x+y$$

【求解】 下面先将该曲面的三维图形绘制出来，如图 10-29 所示。

```
>> [x,y]=meshgrid(-1:0.1:3,-3:0.1:3); z=sin(3*x.*y+2)+x.*y+x+y;
   surf(x,y,z); shading interp  % 用光滑曲面表示目标函数
```

若用传统非线性规划搜索的方法，则可以编写出如下的目标函数表示：

```
function y=c10mga5(x)
y=sin(3*x(1)*x(2)+2)+x(1)*x(2)+x(1)+x(2);
```

这样就可以使用下面的 MATLAB 命令去搜索最优值。从得出的曲面看，该曲面存在很多凹凸不平的地方。如果初值选择不当，很容易陷入局部最小值区域。例如，用户分别选择初值 [1,3] 和 [0,0]，则将得出不同的“最优解”。故对这样的问题，用传统的非线性规划求解算法不易求出全局最优解。

```
>> x0=[1,3]; x=fmincon('c10mga5',x0,[],[],[],[],[-1;-3],[3;3])
```

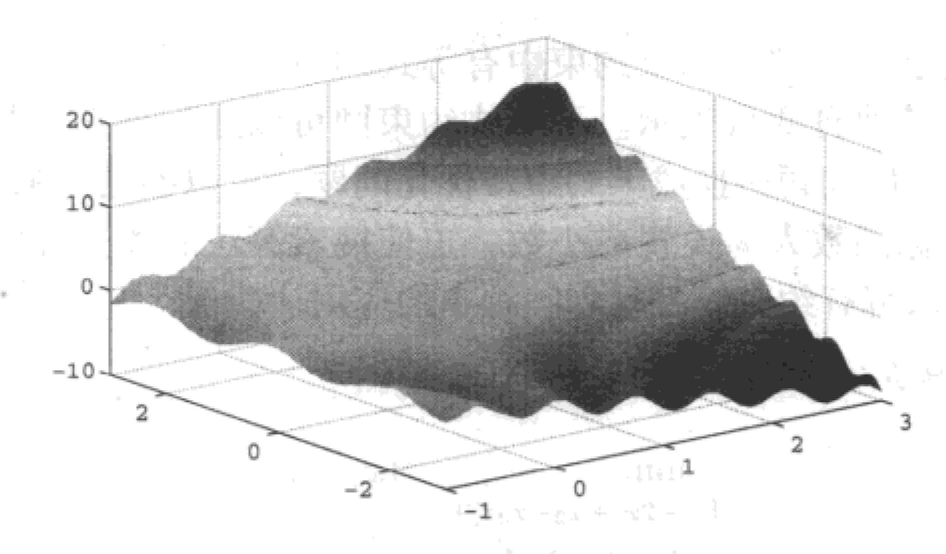


图 10-29 目标函数的三维曲面表示

```
x =
    -1.00000000000000   1.19031289884159
```

现在考虑用遗传算法求解函数 gaopt() 来求取该问题的最优解，可以写出如下的目标函数：

```
function [sol,y]=c10mga6(sol,options)
x=sol(1:2); y=-sin(3*x(1)*x(2)+2)-x(1)*x(2)-x(1)-x(2);
```

指定了搜索的上下界，调用该函数可以立即求出该问题的最优解为 $x = 2.51604948213094$, $y = -3$。遗传算法求解的中间结果在表 10-9 中给出。可以看出，搜索到 40 多代时就已经得出很好的结果了，没有必要非得搜索 500 代。对本例子来说，可以由默认的初始种群计算出全局最优解，而不是局部最优解。

```
>> xmM=[-1 3; -3 3]; [a,b,c,d]=gaopt(xmM,'c10mga6',[],[],[],'maxGenTerm',500)
a =
   2.51604948213094  -3.00000000000000   9.00709500762913
```

表 10-9 遗传算法搜索部分中间结果

代	x	y	$z(x,y)$	代	x	y	$z(x,y)$	代	x	y	$z(x,y)$
1	2.833342	-2.750378	8.285009	11	2.559481	-2.985504	8.94318	21	2.517594	-3	9.007001
3	2.584278	-2.98563	8.864395	12	2.494709	-3	8.988906	24	2.517461	-3	9.007016
6	2.584278	-2.985504	8.864593	14	2.521225	-2.994066	8.996172	27	2.5173	-3	9.007033
8	2.576973	-2.985504	8.89188	15	2.51929	-3	9.006681	30	2.51678	-3	9.007074
9	2.564085	-2.985504	8.931682	18	2.518915	-3	9.006771	46	2.516	-3	9.007095
10	2.494709	-2.985506	8.935845	19	2.51807	-3	9.006934	500	2.516049	-3	9.007095

10.3.3 遗传算法在有约束最优化问题中的应用

前面已经通过例子介绍过，遗传算法可以用于无约束最优化问题的求解。但在实际科学研究中经常需要求解有约束的最优化问题，例如第 6 章中分别介绍了利用传统的数学方法求解线性规划、二次型规划及一般非线性规划的方法。经典的遗传算法不能直接用

于有约束最优化问题的求解。如果约束中有等式约束，则可以通过等式求解的方式将其中若干个自变量用其他自变量表示。不等式约束则可以通过惩罚函数方法转换到目标函数中，这样可以将原始问题转换成无约束最优化问题，可以参见文献 [45]，或直接将不满足等式约束时的目标函数人为设置成小数，迫使搜索算法脱离这样的 x 值。下面通过例子演示遗传算法在求解有约束线性规划问题中的应用。

【例 10-18】 试用遗传算法求解下面的线性规划问题。

$$\min_{x \text{ s.t.} \begin{cases} -2x_1+x_2+x_3\leqslant 9 \\ -x_1+x_2\geqslant -4 \\ 4x_1-2x_2-3x_3=-6 \\ x_{1,2}\leqslant 0, x_3\geqslant 0 \end{cases}} (x_1+2x_2+3x_3)$$

【求解】 由等式约束可以解出 $x_3=(6+4x_1-2x_2)/3$。可见，原来三元最优化问题可以转换成二元最优化问题。由上面的推导，可以用下面的 MATLAB 函数描述目标函数为

```
function [sol,y]=c10mga4(sol,options)
x=sol(1:2); x=x(:); x(3)=(6+4*x(1)-2*x(2))/3;
y1=[-2 1 1]*x; y2=[-1 1 0]*x;
if (y1>9 | y2<-4 | x(3)<0), y=-100; else, y=-[1 2 3]*x; end
```

其中用 x_1,x_2 数据计算出 x_3 的值，并判定约束条件是否满足，在不满足约束条件时人为地将目标函数的值设置成 -100，有意排除这些种群，这样就能由下面语句较好地解决有约束最优化问题。

```
>> [a,b,c]=gaopt([-1000 0; -1000 0],'c10mga4',[],[],[],'maxGenTerm',1000);
   c=[c(1:15:end,:); c(end,:)]; a,c
a =
   -6.99991670058531 -10.99991634912998  28.99958350292656
```

这里只得出了 x_1,x_2，而 x_3 可以由 $x_3=(6+4x_1-2x_2)/3=0.00005529863957$。遗传算法的中间结果由表 10-10 给出。

表 10-10 遗传算法搜索部分中间结果

代	x_1	x_2	$f(x)$	代	x_1	x_2	$f(x)$
1	-269.8650965	-377.0311139	-100	57	-1.45193811	0	1.25969055
90	-1.59065611	-0.6689306314	1.953280549	123	-1.860366329	-0.8104298101	3.301831646
186	-5.897126844	-9.4606018	23.48563422	356	-6.388432983	-10.04209934	25.94216492
613	-6.889962856	-10.8068347	28.44981428	676	-6.902971784	-10.80664291	28.51485892
823	-6.925942985	-10.87574207	28.62971492	1080	-6.963837954	-10.92786854	28.81918977
1397	-6.990366303	-10.98077174	28.95183151	1768	-6.997345129	-10.99471444	28.98672565
1849	-6.997368852	-10.99494982	28.98684426	1952	-6.999503469	-10.99926216	28.99751735
1994	-6.999896151	-10.99985241	28.99948076	2000	-6.999916701	-10.99991635	28.9995835

其实，该问题用线性规划函数可以立即得出更精确的结果。

```
>> f=[1 2 3]; A=[-2 1 1; 1 -1 0]; B=[9; 4]; Aeq=[4 -2 -3]; Beq=-6;
   x=linprog(f,A,B,Aeq,Beq,[-inf;-inf;0],[0;0;inf]); x'
ans =
  -6.99999999999967    -10.99999999999935      0.00000000000000
>> f*x
ans =
 -28.99999999999836
```

从最优化问题求解的方法看，最优化工具箱中的函数一次只能搜索到一个解，对非凸性问题来说往往可能找到一个局部最优值，而用遗传算法则可以同时从一组初值点出发，有可能找到更好的局部最优值甚至全局最优值，但其求取最优值算法的精度和速度均不是很理想。在实际求解问题中，可以考虑采用这样的策略，先用遗传算法初步定出较好最优值所在的大概位置，然后以该位置为初值，调用最优化工具箱中函数快速、准确地求出该最优值。

10.4 小波变换及其在数据处理中的应用

Fourier 变换是信号处理中一种重要的手段，但因其本身的局限性 (例如，它只能将时域波形变换成频域表示，完全失去了与原来时域信号的对应关系)，所以对某些特定的信号进行处理时不是很理想。前面已经介绍过，Fourier 变换将给定的信号展开成不同频率的正弦信号之和。对平稳的信号来说，用 Fourier 变换的方式可以对信号进行较好的分析，但对非平稳的信号和暂变的信号来说，不适合使用 Fourier 变换进行分析。因为 Fourier 变换会略去重要的暂态信息，因此需要引入其他的变换方式解决这样的问题，如短时 Fourier 变换。20 世纪 80 年代逐渐兴起的小波分析技术弥补了这方面的不足，目前正越来越广泛地应用于数据与信号处理以及图像处理等领域。

10.4.1 小波变换及基小波波形

所谓小波 (wavelet)，是指均值为 0 的一类波形。对该波形进行平移和小波分析是将原来的信号分解为基小波波形经过平移与比例变化后的一系列波形。本小节将介绍连续、离散小波变换的基本内容，并介绍几种常用的基小波函数波形。

10.4.1.1 连续小波变换

连续小波变换的变换公式为

$$\mathscr{W}_{a,b}^{\psi}[f(t)]=\frac{1}{\sqrt{|a|}}\int_{-\infty}^{\infty}f(t)\overline{\psi_{a,b}(t)}\mathrm{d}t=W_{\psi}(a,b) \tag{10-4-1}$$

其中，

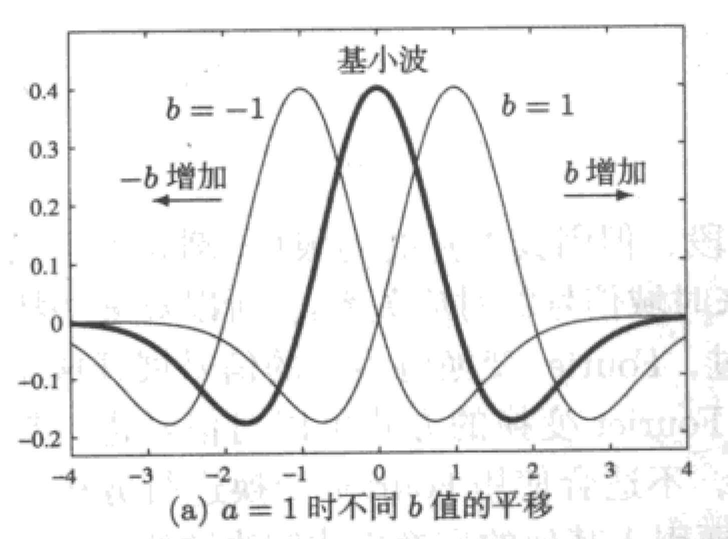
$$\psi_{a,b}(t)=\psi\left(\frac{t-b}{a}\right),\ \text{且}\ \int_{-\infty}^{\infty}\psi(t)\mathrm{d}t=0 \tag{10-4-2}$$

且 $\psi(t)$ 称为基小波，$\psi_{a,b}(t)$ 为基小波通过平移、比例缩放构成的小波信号。

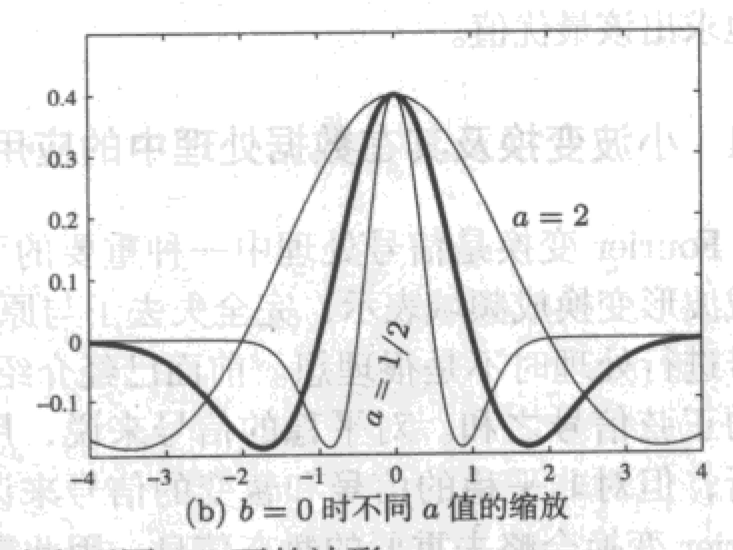
【例 10-19】假设"墨西哥帽"基小波函数由 $\psi(t)=\dfrac{1-t^2}{\sqrt{2\pi}}\mathrm{e}^{-\frac{t^2}{2}}$ 给出，试绘制出不同 a,b 值变换下的小波函数。

【求解】因为基小波函数已给出，故可以用符号运算工具箱表示该函数，并用 ezplot() 函数将其绘制出来。利用符号运算工具箱中给出的 subs() 函数则可以将 t 变量替换成小波函数 $\psi_{a,b}(t)$ 所需的形式，并在原来坐标系下绘制出不同 a,b 参数下的小波函数曲线，分别如图 10-30 (a)、图 10-30 (b) 所示。

```
>> syms t; f=(1-t^2)*exp(-t^2/2)/sqrt(2*pi);
   ezplot(f,-4,4), hold on;  % 绘制基小波，并保护坐标系不被刷新
   ezplot(subs(f,t,t-1),-4,4); ezplot(subs(f,t,t+1),-4,4)  % 小波平移
   figure; ezplot(f,-4,4), hold on;
   ezplot(subs(f,t,t/2),-4,4); ezplot(subs(f,t,2*t),-4,4)  % 小波缩放
```



(a) $a=1$ 时不同 b 值的平移



(b) $b=0$ 时不同 a 值的缩放

图 10-30 墨西哥帽基小波在不同 a,b 下的波形

从上面的例子可以看出，b 参数将向左右方向平移基小波信号，a 参数将起到扩展或压缩基小波的作用。若 $a<1$，将压缩基小波信号的宽度，形成新的小波信号。小波分析是对各个 a,b 组合计算出系数，然后将这些小波信号乘以相应的系数再叠加起来，重构出原信号。和其他积分变换一样，重构原信号又称为小波反变换，或小波反演。

小波反变换的定义为

$$f(t)=\frac{1}{C_\psi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}W_\psi(a,b)\psi_{a,b}(t)\mathrm{d}a\mathrm{d}b \tag{10-4-3}$$

其中，

$$C_\psi=\int_{-\infty}^{\infty}\frac{|\hat{\psi}(\omega)|^2}{|\omega|}\mathrm{d}\omega \tag{10-4-4}$$

用连续小波变换的系数计算可以由 cwt() 函数完成。该函数的调用格式为

Z=cwt(y,a,基小波名称)	计算小波系数矩阵 Z
Z=cwt(y,a,基小波名称,'plot')	直接绘制小波系数绝对值图

其中，基小波名称在后面将详细介绍，这里只使用前面定义的墨西哥帽函数，其名称标记为 'mexh'。

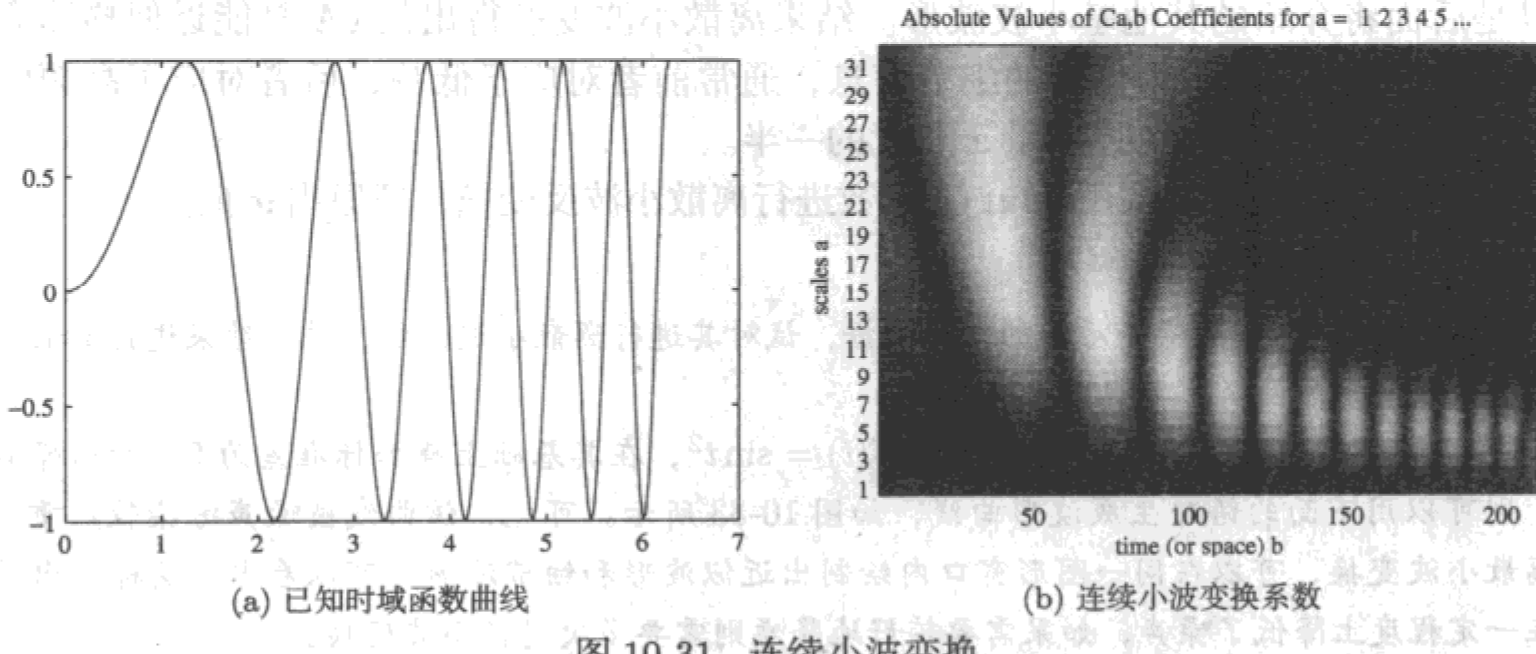
【例 10-20】 试对信号 $f(t)=\sin t^2$ 进行连续小波分解，并绘制出其系数图。

【求解】 可以由下面的语句直接生成 $t\in[0,2\pi]$ 区间内的数据，并绘制出时域数据曲线，如图 10-31 (a) 所示。

```
>> t=0:0.03:2*pi; y=sin(t.^2); plot(t,y)
```

选择 'mexh' 基小波作为模板，则可以绘制出小波系数 $W_\psi(a,b)$ 的图形，如图 10-31 (b) 所示。

```
>> a=1:32; Z=cwt(y,a,'mexh','plot');  % 绘制绝对值图
```



(a) 已知时域函数曲线　　(b) 连续小波变换系数

图 10-31　连续小波变换

还可以用下面的命令绘制出小波系数的三维表面图，如图 10-32 所示。

```
>> surf(t,a,Z); shading flat; axis([0 2*pi,0,32,min(Z(:)) max(Z(:))])
```

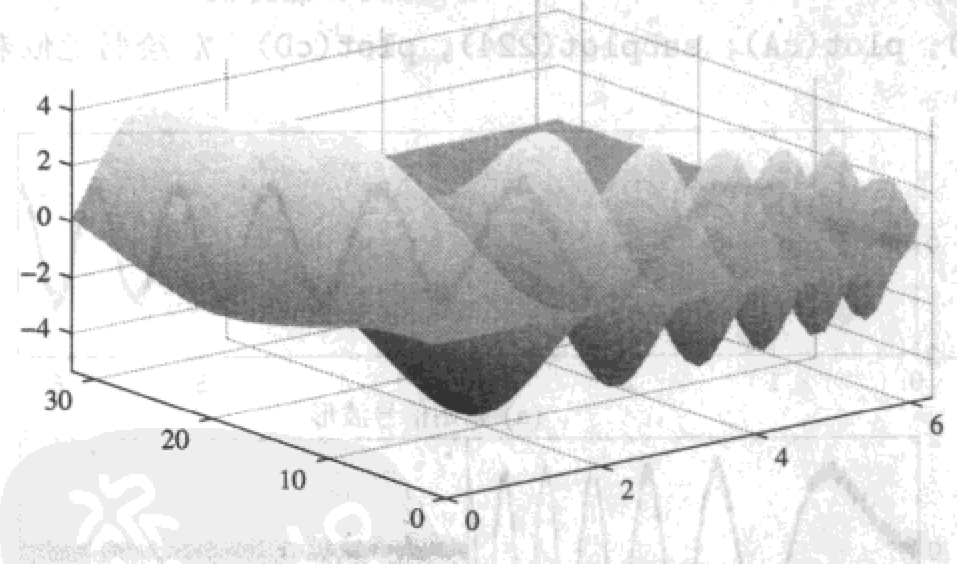


图 10-32　连续小波变换系数的三维表面图表示

10.4.1.2　离散小波变换

若 $f(t)$ 信号取其离散值 $f(k)$，且选择基小波函数 $\psi(t)$，则结果平移与缩放的小波函数为 $\psi_{a,b}(t)=\sqrt{2}\psi\left(2^m t-n\right)$，其离散形式可以写成 $\psi_{a,b}(k)=\sqrt{2}\psi\left(2^m k-n\right)$，这时可以定义出离散信号的小波变换为

$$\mathscr{W}_{n,m}^{\psi}[f(k)]=\sqrt{2}\sum_k f(k)\overline{\psi\left(2^m k-n\right)}\mathrm{d}t=W_\psi(m,n) \tag{10-4-5}$$

离散小波反演公式为

$$f(k) = \sum_m \sum_n W_{n,m}(k)\psi_{m,n}(k) \tag{10-4-6}$$

MATLAB 的小波工具箱中给出了 dwt() 函数，可以对给定的数据进行一次离散小波变换。该函数的调用格式为

```
[cA,cD]=dwt(x,fun)
```

其中，x 为原始数据，fun 为选择的基小波函数名，可以为前面介绍的 'mexh' 函数，还可以是后面将介绍的其他基小波波形。结果离散小波变换得出的 cA 是能近似描述原波形的小波系数，而 cD 为信号的细节信息，通常前者对应于低频，后者对应于高频的部分。cA 和 cD 的长度均为原向量 x 长度的一半。

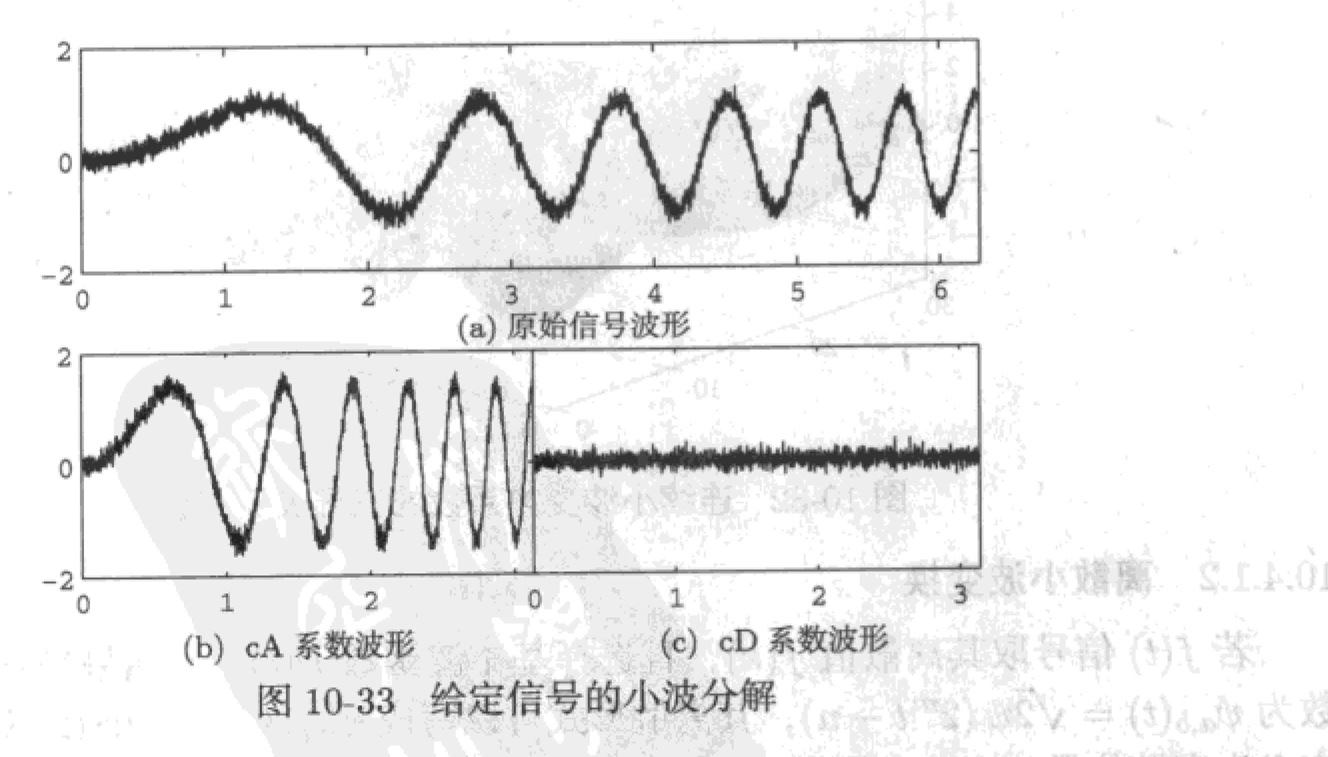
由 cA 和 cD 还可以调用 idwt() 函数进行离散小波反变换，还原出 $\hat{x}$ 向量。

```
x̂=idwt(cA,cD,fun)
```

【例 10-21】 生成一组被噪声污染的信号数据，试对其进行离散小波分解，并对结果进行离散小波反变换，反演出原函数，再观察反演结果。

【求解】 仿照例 10-20 中给出的信号模型 $f(t) = \sin t^2$，在其基础上叠加标准差为 0.1 的白噪声信号，则可以用下面的语句生成波形曲线，如图 10-33 所示。可见，该曲线被噪声污染较严重。通过离散小波变换，可以在同一图形窗口内绘制出近似波形和细节波形。可以看出，这样得出的波形在一定程度上降低了噪声，如果需要较好地降噪则需要多次进行小波变换。

```
>> x=0:0.002:2*pi; y=sin(x.^2); r=0.1*randn(size(x));
   y1=y+r; subplot(211); plot(x,y1)   % 绘制原始信号
   [cA,cD]=dwt(y1,'db4');             % 离散小波变换
   subplot(223), plot(cA); subplot(224), plot(cD)  % 绘制近似和细节信号
```



(a) 原始信号波形

(b) cA 系数波形　　(c) cD 系数波形

图 10-33　给定信号的小波分解

对得出的 cA 和 cD 向量还可以进行离散小波反变换。经过和原信号比较，得出的信号基本上还原了原始信号，误差极小。

```
>> y2=idwt(cA,cD,'db4'); norm(y1-y2)  % 检验反变换对信号的还原程度
```

```
ans =
    6.183804676580192e-012
```

10.4.1.3 小波工具箱中提供的基小波函数

小波工具箱中提供了大量的基小波模板，如 Haar 小波、Daubechies 族小波、墨西哥帽小波、Bior 族小波等，可以直接调用。用 `wavemngr()` 函数即可列出允许使用的基小波名称。该函数可以由下面的格式调用：

```
wavemngr('read',1)
```

例如，Haar 小波可以选择名称 'haar'，Daubechies 族小波可以有 'db1'、'db2' 等模板，Bior 族小波可以有 'bior1.3'、'bior2.4' 等，墨西哥帽小波可以选择 'mexh' 等。小波模板数据可以由 `wavefun()` 来计算。该函数的调用格式为

$[\psi, x]$=wavefun(fun,n)，Gauss、墨西哥帽等基小波函数

$[\phi, \psi, x]$=wavefun(fun,n)，Daubechies 族、Symlets 族等正交基小波函数

$[\phi_1, \psi_1, \phi_2, \psi_2, x]$=wavefun(fun,$n$)，Bior 族等基小波函数

其中，n 为迭代次数，其默认值为 8。ψ 为基小波，ϕ 为小波导数，而 Bior 小波中 ϕ,ψ 向量的下标 1 表示用于小波分解，2 用于小波重建。

【例 10-22】 选择 Daubechies 6 小波 ('db6')，试绘制出不同阶次下的基小波波形。

【求解】 用下面的语句可以立即绘制出该小波在 2,4,6,8 迭代次数下的波形，如图 10-34 所示。可见，当迭代次数选为 8 时能得出相当平滑的波形，所以一般可以选择的迭代次数为 8。

```
>> [a,y,x]=wavefun('db6',2); subplot(141), plot(x,y)
   [a,y,x]=wavefun('db6',4); subplot(142), plot(x,y)
   [a,y,x]=wavefun('db6',6); subplot(143), plot(x,y)
   [a,y,x]=wavefun('db6',8); subplot(144), plot(x,y)
```

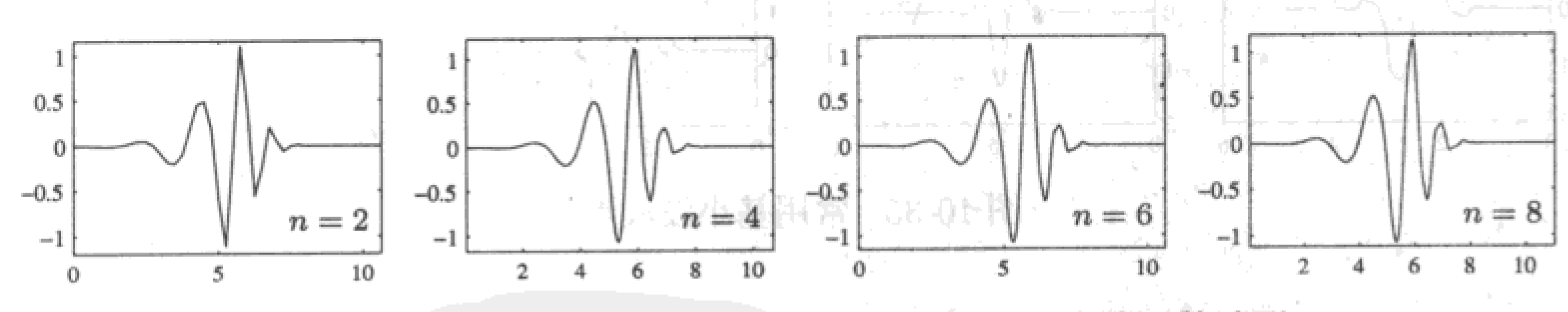
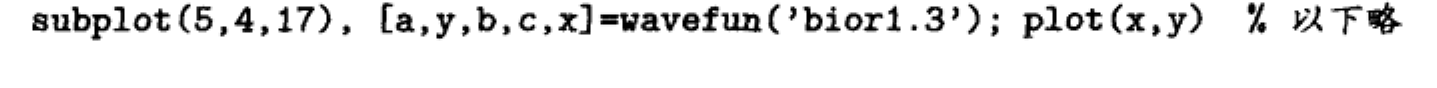


图 10-34 不同迭代次数下的 Daubechies 6 基小波函数波形

【例 10-23】 试绘制出常用基小波波形。

【求解】 下面的语句可以绘制出一些常用的基小波波形，如图 10-35 所示。其中，'db1' 小波实际上就是 Haar 小波。还可以看出，Daubechies 族小波在 'db6' 取值时较平滑，Symlets 族小波在 'sym6' 时较平滑。

```
>> subplot(5,4,1), [a,y,x]=wavefun('db1'); plot(x,y), % 同样绘制其他 db 小波
   subplot(5,4,9), [a,y,x]=wavefun('sym2'); plot(x,y), % 同样绘制其他 sym 小波
   subplot(5,4,13), [a,y,x]=wavefun('coif2'); plot(x,y)
   subplot(5,4,15), [y,x]=wavefun('gaus2'); plot(x,y)
```



```
subplot(5,4,17), [a,y,b,c,x]=wavefun('bior1.3'); plot(x,y)  % 以下略
```

图 10-35 常用基小波波形

10.4.2 小波变换技术在信号处理中的应用

与 Fourier 变换技术以及基于该技术的频域方法类似，小波技术也可以用于信号处理和二维信号处理 (如图像处理)，且可以显示出传统频域分析方法难以实现的特性。本节将介绍基于小波变换技术的信号分解与重建方法及 MATLAB 实现，并将通过信号噪声过滤的例子来演示小波在降噪中的应用。

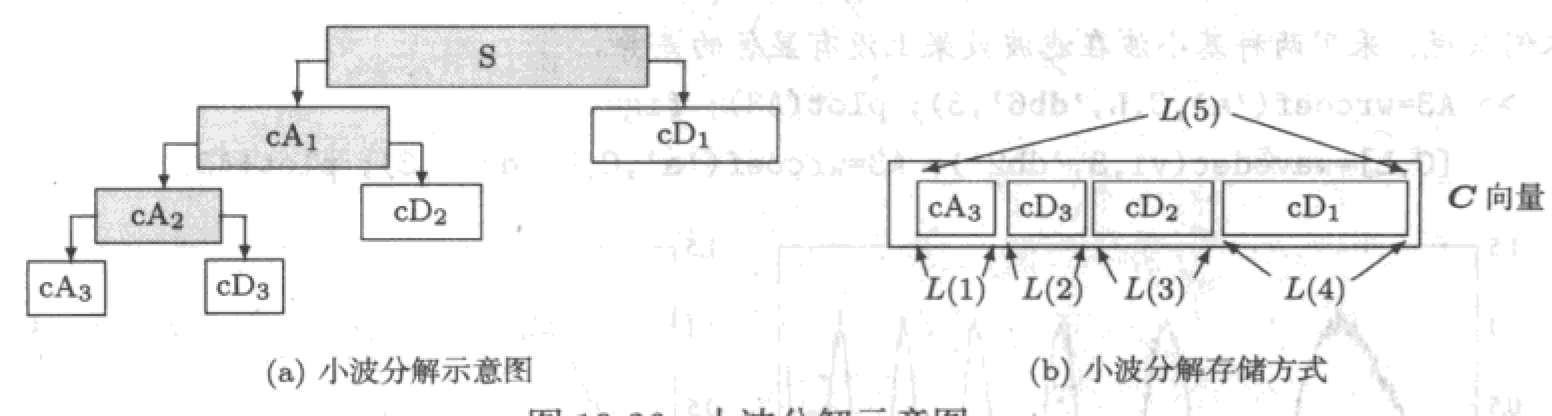
通过小波变换方法对某给定信号进行分解，可以将给定信号 S 分解成两个部分，即 cA_1 和 cD_1，这时得出的 cA_1 和 cD_1 信号的数据量均为原数据 S 的一半，且 cA_1 保留原信号的低频信息或近似信息，而 cD_1 保留该信号的高频信息或细节信息。从信号的噪声过滤的角度看，cA_1 信号有效的成分多，而 cD_1 多属于噪声信号。对 cA_1 信号再进行一

步小波分解，则得出 cA_2, cD_2。对 cA_2 再进行分解则得出 cA_3 和 cD_3，如此还可以再进行多步分解，分解的过程如图 10-36 (a) 所示。和前面介绍的单级小波分解类似，各个 cA 序列称为近似系数，而 cD 段称为细节系数。

MATLAB 的小波分析工具箱提供了 wavedec() 函数，可以用于一维信号的小波分解。该函数的调用格式为

$[\boldsymbol{C},\boldsymbol{L}]$=wavedec($\boldsymbol{x}$,$n$,fun)

其中，$\boldsymbol{x}$ 为原始信号，n 为分解的步数，如取 $n=3$，fun 为所选基小波的名称，如 'db6'，分解后可以得出 $\boldsymbol{C}$ 和 $\boldsymbol{L}$ 两个向量，其组成形式如图 10-36 (b) 所示，即 $\boldsymbol{C}$ 向量是按照如图 10-36 (b) 所示的顺序将这些段短向量接成的和 $\boldsymbol{x}$ 等长度的向量，且每个子段的长度由 $\boldsymbol{L}$ 向量相应元素给出。



(a) 小波分解示意图　　(b) 小波分解存储方式

图 10-36 小波分解示意图

由分解后的 $\boldsymbol{C}$ 和 $\boldsymbol{L}$ 向量提取近似系数 cA 和细节系数 cD 可以分别由 appcoef() 和 detcoef() 函数实现。其调用格式分别为

cA_n=appcoef($\boldsymbol{C}$,$\boldsymbol{L}$,fun,n);　提取近似系数
cD_i=detcoef($\boldsymbol{C}$,$\boldsymbol{L}$,i);　提取第 i 段细节系数

由得出的近似系数和细节系数重建原信号则可以略去部分噪声信息，信号重建的函数可以使用 wrcoef()。该函数的调用格式为

$\hat{\boldsymbol{x}}$=wrcoef(类型,$\boldsymbol{C}$,$\boldsymbol{L}$,fun,n)

其中，“类型”可以选择为 'a' 和 'd'，用来确定是利用近似小波系数还是利用细节系数来进行原信号重建，若选择了近似系数，则可以较好地解决信号降噪问题。

【例 10-24】 对例 10-21 中的数据进行 3 次小波分解，试用各种基小波函数对其进行降噪处理，比较降噪效果。

【求解】 由例 10-21 中给出的数据信号可以绘制出如图 10-33 (a) 所示的原信号波形曲线。

```
>> x=0:0.002:2*pi; y=sin(x.^2); r=0.1*randn(size(x)); y1=y+r; plot(x,y1)
```

对给定的数据进行 3 次小波分解，则可以用下面的语句绘制出相关各个子信号的波形，如图 10-37 所示，其中作者根据需要对各个坐标系的宽度进行了手工调整。可见，每次分解都能滤去一部分噪声，故最终得出的 cA_3 含有的噪声成分较少。

```
>> [C,L]=wavedec(y1,3,'db6');
   cA3=C(1:L(1)); subplot(141), plot(cA3)
   dA3=C(L(1)+1:sum(L([1 2]))); subplot(142), plot(dA3)
```

```
dA2=C(sum(L(1:2))+1:sum(L(1:3))); subplot(143), plot(dA2)
dA1=C(sum(L(1:3))+1:sum(L(1:4))); subplot(144), plot(dA1)
```

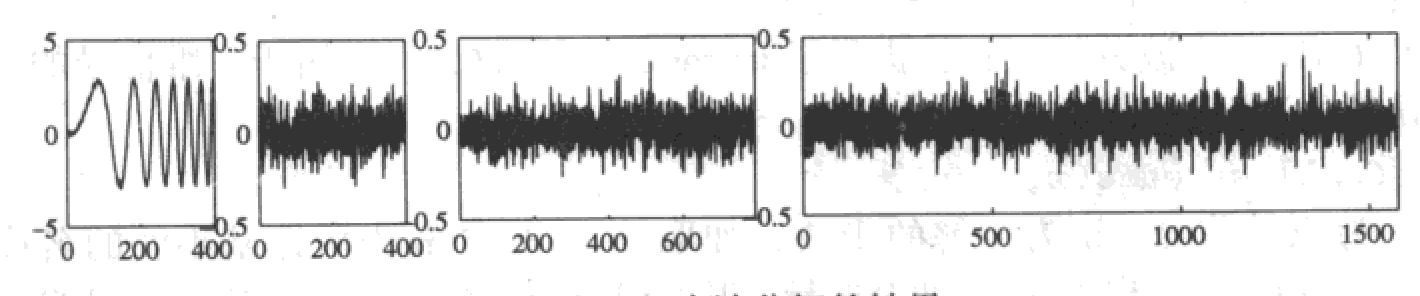
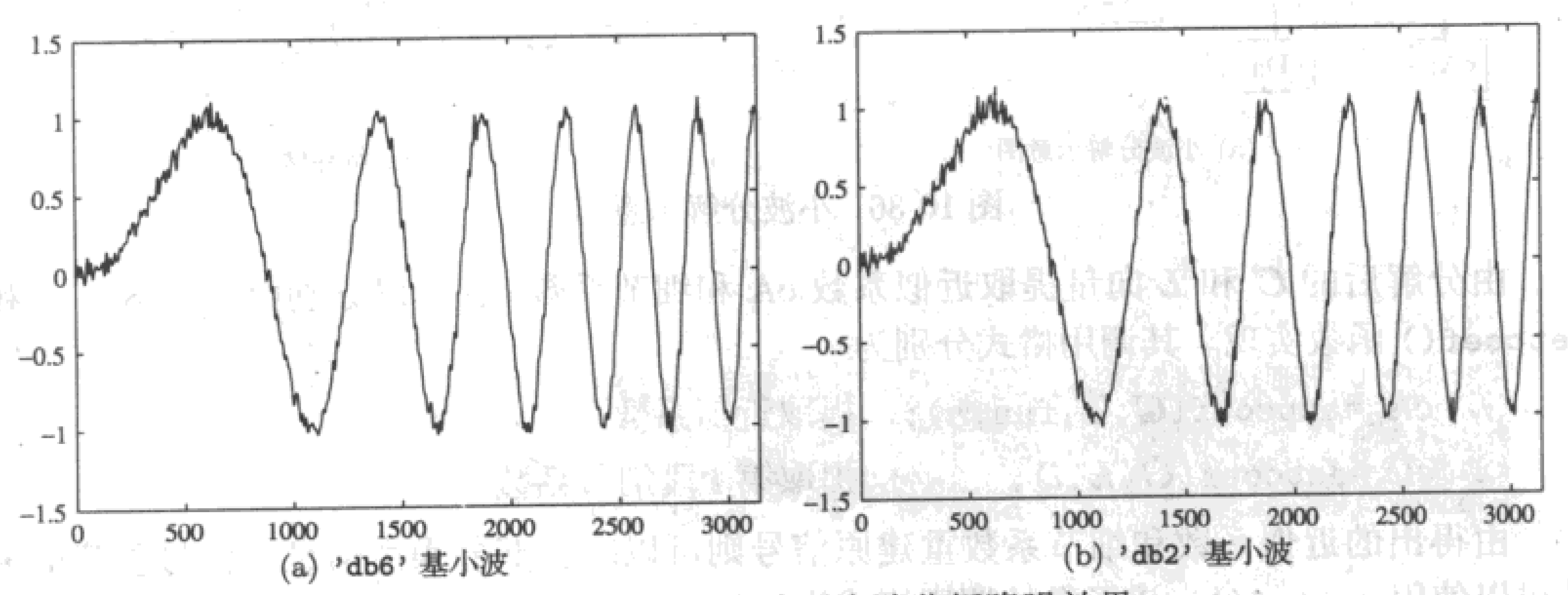


图 10-37 小波分解的结果

现在仍采用 'db6' 基小波，则可以由下面的语句绘制出滤波后的近似波形，如图 10-38 (a) 所示。若采用 'db2' 基小波，则可以得出如图 10-38 (b) 所示的滤波近似波形。从得出的结果看，对本例来说，采用两种基小波在滤波效果上没有显著的差异。

```
>> A3=wrcoef('a',C,L,'db6',3); plot(A3); figure
   [C,L]=wavedec(y1,3,'db2'); A3=wrcoef('a',C,L,'db2',3); plot(A3)
```



(a) 'db6' 基小波 (b) 'db2' 基小波

图 10-38 不同基小波下小波分析降噪效果

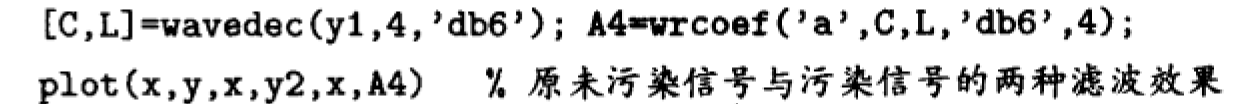
其实，用下面的语句还可以得出另外两种常用基小波下降噪的效果，和如图 10-38 (a)、图 10-38 (b) 所示的结果很接近。故对本例来说，降噪效果仍无显著差异。

```
>> [C,L]=wavedec(y1,3,'bior2.6'); A3=wrcoef('a',C,L,'bior2.6',3); plot(A3)
   [C,L]=wavedec(y1,3,'coif4'); A3=wrcoef('a',C,L,'coif4',3); plot(A3)
```

【例 10-25】重新考虑例 8-32 中的数字滤波问题，试用小波对其进行滤波，并比较滤波效果。

【求解】用下面语句可以重复例 8-32 中给出的滤波器滤波结果，同样，采用 4 级 'db6' 小波，也可以降噪效果，这些降噪效果在图 10-39 中给出，图中有延迟的曲线为第 8 章的滤波方法得出的。可见，小波降噪方法不会产生数字滤波器那样的时间延迟，滤波效果也明显好于滤波器。

```
>> b=1.2296e-6*conv([1 4 6 4 1],[1 3 3 1]); a=conv([1,-0.7265],...
      conv([1,-1.488,0.5644],conv([1,-1.595,0.6769],[1,-1.78,0.8713])));
   x=0:0.002:2; y=exp(-x).*sin(5*x); r=0.05*randn(size(x)); y1=y+r;
   y2=filter(b,a,y1); % 例 8-32 中给出的滤波方法
```



```
[C,L]=wavedec(y1,4,'db6'); A4=wrcoef('a',C,L,'db6',4);
plot(x,y,x,y2,x,A4)    % 原来污染信号与污染信号的两种滤波效果
```

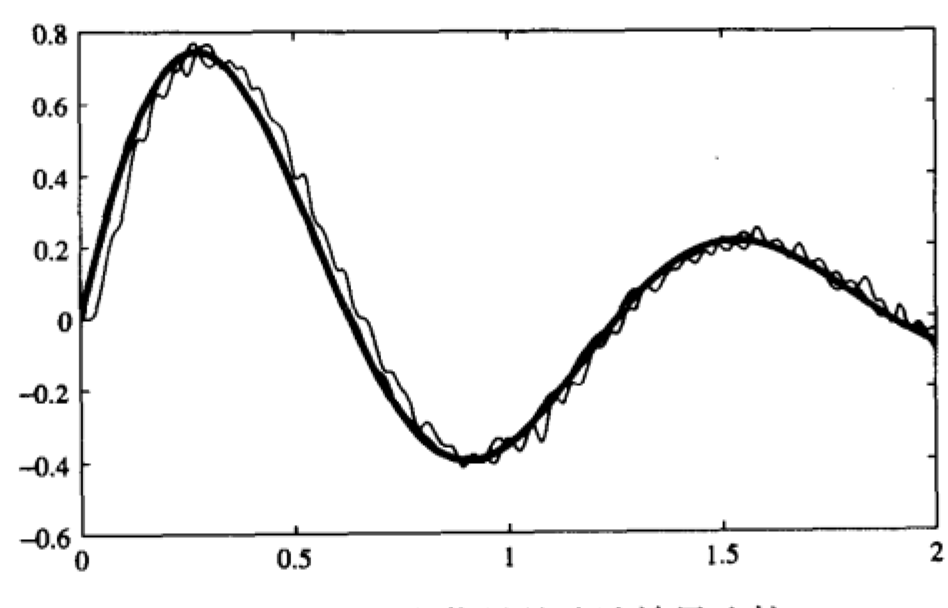


图 10-39 给定信号的滤波效果比较

10.4.3 小波问题的程序界面

小波工具箱还提供了求解一维、二维小波变换问题的图形用户界面。在 MATLAB 命令窗口中输入 wavemenu 命令可以启动该程序，得出如图 10-40 所示的图形界面。如果想解决一维小波变换问题，则单击 Wavelet 1-D (一维小波分析) 按钮，该程序将引导用户输入数据文件，选择小波并进行小波分析的全过程。该界面使用起来较容易，故不在这里详细介绍了。

10.5 粗糙集理论与应用

10.5.1 粗糙集理论介绍

10.5.1.1 粗糙集理论简介

粗糙集 (rough set) 是波兰数学家 Z Pawlak 为开发自动规则生成系统及研究软计算问题于 1982 年提出的。19 世纪 90 年代初，人们才逐渐认识到粗糙集的重要性。1991 年 Z Pawlak 出版了专著，奠定了严密的数学基础[36]。基于粗糙集的知识理论由于不需要预先给定某些特征或属性的数量，可从现有的数据出发给出知识的简化和相对简化、范畴的简化和相对简化方法，为处理不精确、不完全信息提供一种更符合人类认知的知识理论。粗糙集理论是一种处理不精确、不确定与不完全数据的新的数学方法。它能有效地分析和处理不精确、不一致、不完整等各种不完备信息，并从中发现隐含的知识，揭示潜在的规律。由于它在机器学习与知识发现、数据挖掘、决策支持与分析、专家系统、归纳推理、模式识别等方面的应用突出，现已成为一个热门的研究领域。

当前国内外学者已公认，该理论是研究数据挖掘、知识约简、信息计算的理论基础，是当前国内外计算机及相关专业的学者和科技人员的研究热点。

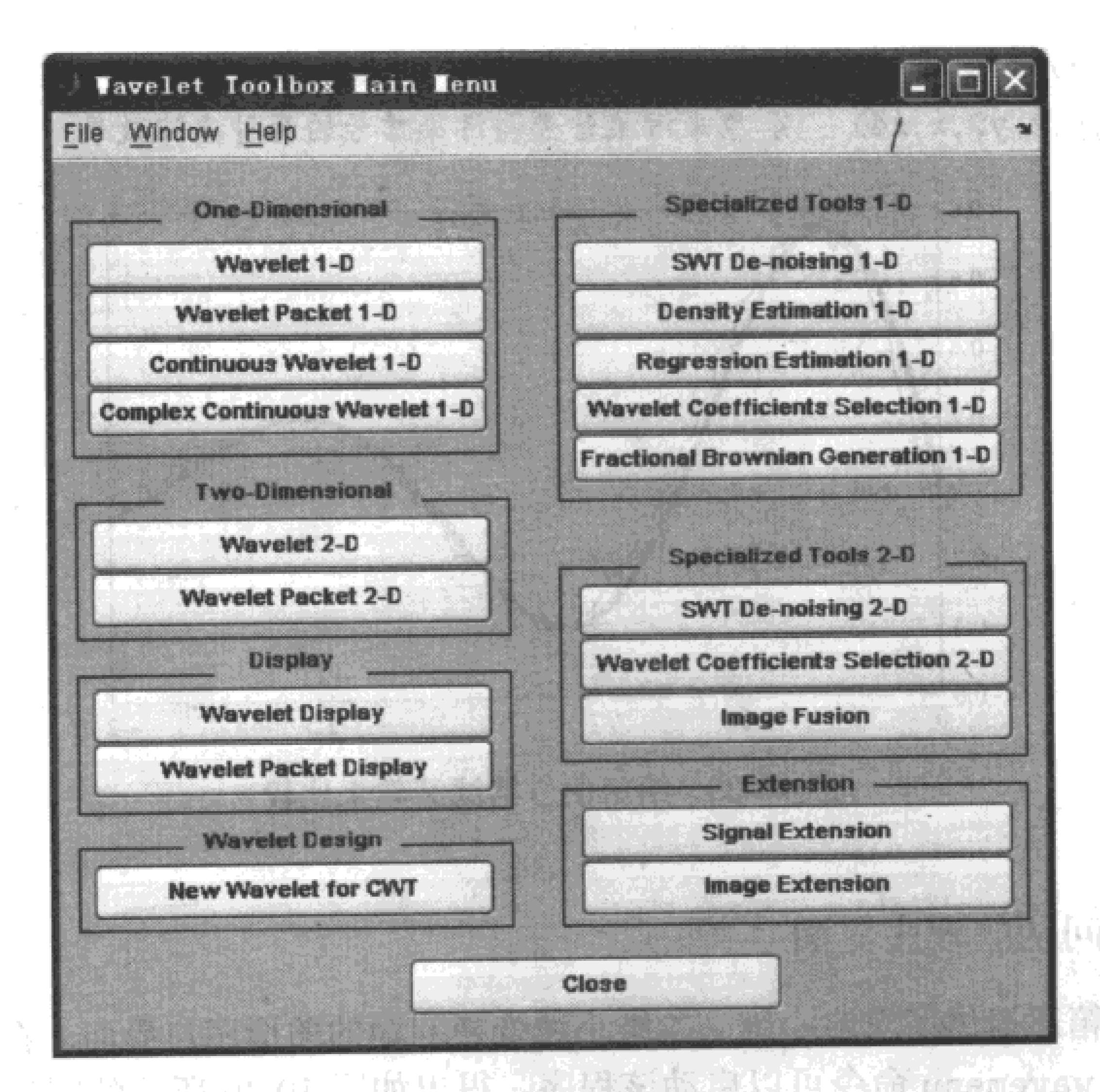


图 10-40 小波分析程序界面

10.5.1.2 粗糙集的基本概念

设 $X,Y \in U, R$ 是定义在 U 上的等价关系，则集合 X 关于 R 的下近似集定义为

$$\underline{\mathscr{R}}(X) = \bigcup\{Y \in U/R:\ Y \subseteq X\} \tag{10-5-1}$$

其中，$\underline{\mathscr{R}}(X)$ 是根据现有知识判断肯定属于 X 的对象组成的最大的集合，称为正区，记为 POS(X)。

类似地，也可以定义出集合 X 关于 R 的上近似集为

$$\overline{\mathscr{R}}(X) = \bigcup\{Y \in U/R:\ Y \cap R \neq \phi\} \tag{10-5-2}$$

其中，ϕ 表示为空集。$\overline{\mathscr{R}}(X)$ 是由所有集合 X 相交非空的等效类的并集，是那些可能属于 X 的对象组成的最小集合。

由上面的定义可以再给出集合边界区定义为 $\mathrm{Bnd}(X) = \overline{\mathscr{R}}(X) - \underline{\mathscr{R}}(X)$。如果 $\mathrm{Bnd}(X)$ 是空集，则称 X 关于 R 是清晰的；反之，如果 $\mathrm{Bnd}(X)$ 非空，则称 X 为关于 R 的粗糙集。

【例 10-26】假设玩具积木集合 $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$，具有“颜色 R_1”、“形状 R_2”、“体积 R_3”这 3 种属性，且 $R_1 = \{0,1,2\}$，分别对应红色、黄色和绿色，“形状”的属性值取为 $R_2 = \{0,1,2\}$，分别对应方、圆、三角形。“体积”的属性关系，又可以取为 $R_3 = \{0,1\}$，分别对应于“大的物体”和“小的物体”。

按照这样的属性关系，假设红色的积木有 $\{x_1,x_2,x_7\}$，绿色的有 $\{x_3,x_4\}$，蓝色的有 $\{x_5,x_6\}$，则可以写出 $U|R_1=\{\{x_1,x_2,x_7\},\{x_3,x_4\},\{x_5,x_6\}\}$。

10.5.1.3 信息决策系统

信息决策系统 T 可以表示为 $T=(U,A,C,D)$，其中，U 是对象的集合，即论域，A 是属性集合，如果属性集 A 可以分为条件属性集 C 和决策属性集 D，即 $C\cup D=A$，$C\cap D=\phi$，则该信息系统称为决策系统或决策表。

粗糙集理论中使用决策表来描述论域中对象。它是一张二维表格，每一行描述一个对象，每一列描述对象的一种属性。属性分为条件属性和决策属性，论域中的对象根据条件属性的不同，被划分到具有不同决策属性的决策类。表 10-11 为一张信息系统决策表的例子，$U=\{x_1,x_2,\cdots\}$ 为对象集，$C=\{s_1,\cdots,s_m\}$ 为条件属性集，$D=\{d_1,\cdots,d_k\}$ 为决策属性集，f_{ij} 表示第 i 个对象的第 j 个条件属性值，g_{ij}是第 i 个对象的第 j 决策条件的属性值。

表 10-11 信息系统决策表

论域	C				D		
U	s_1	s_2	$\cdots$	s_m	d_1	$\cdots$	d_k
x_1	f_{11}	f_{12}	$\cdots$	f_{1m}	g_{11}	$\cdots$	g_{1k}
x_2	f_{21}	f_{22}	$\cdots$	f_{2m}	g_{21}	$\cdots$	g_{2k}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_n	f_{n1}	f_{n2}	$\cdots$	f_{nm}	g_{n1}	$\cdots$	g_{nk}

表 10-12 例 10-27 信息决策系统表

论域	C 属性				D
U	颜色 a	形状 b	大小 c	价位 d	销量
1	1	0	1	1	1
2	1	0	0	0	1
3	0	0	1	0	0
4	1	1	0	1	0
5	1	1	1	2	2
6	2	1	0	2	2
7	2	2	0	2	2

【例 10-27】 考虑例 10-26 中给出的玩具积木论域集合，假设有 4 个相关属性，颜色属性 a、形状属性 b、大小属性 c 和价位属性 d，且已知 $x_{1,2,4,5}$ 为黄色的，x_3 为红色的，$x_{6,7}$ 为绿色的；$x_{1,2,3}$ 为方形的，$x_{4,5,6}$ 为圆形的，x_7 为三角形的；$x_{1,3,5}$ 为大玩具，其余为小玩具；$x_{2,3}$ 价位较低，$x_{1,4}$ 价位中等，$x_{5,6,7}$ 价位较高。从实际销售情况看，$x_{3,4}$ 销售很好，$x_{1,2}$ 销售一般，而 $x_{5,6,7}$ 销售较差，试列出信息决策系统表。

【求解】 设颜色属性中用 $\{0,1,2\}$ 分别表示红色、黄色和绿色，则形状属性中 $\{0,1,2\}$ 分别表示方形、圆形和三角形，大小属性中 $\{0,1\}$ 分别表示玩具的小和大，价位属性中 $\{0,1,2\}$ 分别表示较低、中等和较高，在决策属性中 $\{0,1,2\}$ 分别表示销售较好、中等和较差，那么根据给出的表格可以立即得出信息决策系统表，如表 10-12 所示。粗糙集理论的一个重要应用是对已知各个条件进行约简，找出哪些属性对玩具销售情况影响大，哪些没有影响。

信息系统决策表在 MATLAB 下可以由一个矩阵 $\boldsymbol{S}$ 来表示，其中前面各列表示 C 属性，后面各列表示 D 属性，这时上近似集 $\overline{\mathscr{R}}(X)$ 和下近似集 $\underline{\mathscr{R}}(X)$ 可以分别由自编的 `rslower()` 和 `rsupper()` 函数求出。它们的调用格式为

S_l=`rslower(`X,$\boldsymbol{a}$,$\boldsymbol{S}$`)`	求出下近似集 S_l
S_u=`rsupper(`X,$\boldsymbol{a}$,$\boldsymbol{S}$`)`	求出上近似集 S_u
S_d=`setdiff(`S_u,S_l`)`	利用 MATLAB 的差集函数可以求出边界集 S_d

【例 10-28】假设论域 $U=\{x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,x_{10}\}$，关系为 $R=\{R_1,R_2\}$，且

$U/R_1=\{\{x_1,x_2,x_3,x_4\},\{x_5,x_6,x_7,x_8\},\{x_9,x_{10}\}\}$,

$U/R_2=\{\{x_1,x_2,x_3\},\{x_4,x_5,x_6,x_7\},\{x_8,x_9,x_{10}\}\}$,

若 $X=\{x_1,x_2,x_3,x_4,x_5\}$，试求出集合 X 的上近似集和下近似集。

【求解】根据题中已知条件，分别简记 U/R_1 和 U/R_2 中的 3 个子集为 $\{0,1,2\}$，则可以建立起表 10-13 中给出的信息系统决策表 $\boldsymbol{S}$。注意，这里为排版方便起见，将信息系统决策表进行了旋转。选择集合 $X=\{1,2,3,4,5\}$，并选择决策表中的第 1,2 列构成 $\boldsymbol{a}$ 向量，则可以由下面语句计算出集合 X 的上下近似集。

表 10-13 信息系统决策表

论域 X	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
U/R_1 关系	0	0	0	0	1	1	1	1	2	2
U/R_2 关系	0	0	0	1	1	1	1	2	2	2

```
>> S=[0,0; 0,0; 0,0; 0,1; 1,1; 1,1; 1,1; 1,2; 2,2; 2,2];
   X=[1,2,3,4,5]; a=[1,2]; S1=rslower(X,a,S)  % 下近似集
S1 =
     1     2     3     4
>> S2=rsupper(X,a,S)   % 上近似集
S2 =
     1     2     3     4     5     6     7
>> Sd=setdiff(S2,S1)   % 边界集
ans =
     5     6     7
```

从决策表可见，$\{U/R_1,U/R_2\}$ 构成的集合总共有 $\{0,0\},\{0,1\},\{1,1\},\{1,2\},\{2,2\}$，选择的样本 $X=\{1,2,3,4,5\}$，涉及到的 $\{U/R_1,U/R_2\}$ 集合只有 $\{0,0\}$、$\{0,1\}$ 和 $\{1,1\}$，所以，肯定属于样本集合 X 的只有 $\{1,2,3,4\}$，亦即 $\{x_1,x_2,x_3,x_4\}$，因为和 x_5 一样具有映射关系 $\{1,1\}$ 的还有 x_6,x_7，所以可以得出下近似集为 $\{x_1,x_2,x_3,x_4\}$。类似地，可能属于样本集合 X 的样本是 X 的上近似集，由于 x_6,x_7 和 X 集合中的 x_5 都为 $\{1,1\}$，所以上近似集中除了下近似集中的样本外还应该包括 x_5,x_6,x_7，这 3 个样本亦为边界集。此外，由于边界集非空，所以属于粗糙集。

在信息系统中，对于每个属性子集 $R\subseteq A$，不可分辨关系为

$$\mathrm{Ind}(R)=\{(x,y)\in U\times U:\ r\in R:\ r(x)=r(y)\} \tag{10-5-3}$$

显然，$\mathrm{Ind}(R)$ 是一个等价关系，在不产生混淆的情况下可以用 R 代替 $\mathrm{Ind}(R)$。

10.5.2 粗糙集数据处理问题的 MATLAB 求解

10.5.2.1 利用粗糙集理论的约简

目前社会已经进入信息时代，人们获得信息越来越容易。但大量的未处理的信息使

我们陷入“数据灾难”、“决策灾难”，导致要么“疲于应付”，要么“弃之不理”。对解决这类问题的研究，一般称为“从数据库中发现知识”与“数据挖掘”[16]。

信息系统约简主要是使信息量减少，它将一些无关或多余的信息忽略掉，而不影响其原有的决策功能。无疑可以设想将约简后的信息重新组合而产生新的决策规则，这类决策规则的前题信息和结论信息可能不同于约简前的任何一条决策规则，但它们能经推理而得到相同或相近的结果。因此这样的研究成果对数据挖掘以及数据库的进一步应用将产生新的影响。

所谓约简，即不含多余属性并保证分类正确的最小条件属性集。一个信息决策表可能同时存在几个约简。关系等价族 R 中所有不可约去的关系称为核，由它构成的集合称为 R 的核集，记成 Core(R)。

这里不详细介绍约简的具体算法，只介绍依据约简算法编写的几个 MATLAB 函数，如 `redu()`、`core()` 等。关于约简算法的详细讨论请见文献 [55]。

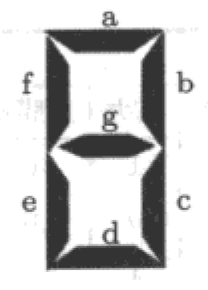
假设信息系统决策表由矩阵 $\boldsymbol{S}$ 表示，其中，第 $\boldsymbol{c}$ 列 C 属性中，$\boldsymbol{d}$ 列为决策列 D 属性中，则从 C 属性中相关的列中直接使得决策列 D 的最少列的求取可以由约简函数 `redu()` 找出。这些函数的调用格式为

$\boldsymbol{y}$=redu($\boldsymbol{c},\boldsymbol{d},\boldsymbol{S}$)　条件约简，找出从 C 属性中选定条件推出 D 的最小集合
$\boldsymbol{y}$=core($\boldsymbol{c},\boldsymbol{d},\boldsymbol{S}$)　求取从 C 属性中选定条件推出 D 的核集

10.5.2.2 粗糙集理论在信息约简中的应用举例

前面介绍了约简的基本概念和约简问题的 MATLAB 求解函数，这些函数在本书配套光盘上给出，可以直接使用。本小节将通过两个实际应用的例子[55]来演示基于粗糙集运算的信息约简方法及 MATLAB 求解。

【例 10-29】0~9 这 10 个数字的显示一般采用 7 段数码管来实现，这 7 段数码管排序如图 10-41 (a) 所示。其中某段数码管发光则记为 1，否则记为 0，这样每个数字显示对应的真值表如表



(a) 7 段数码管

(b) 约简示意图

图 10-41 数码管显示及数码管约简结果

10-14 所示。试用粗糙集的方法对其进行约简，找出不必要的数码管。

【求解】从人类对数字的直观理解看，这 7 段数码管当然全是必要的，缺少哪段都不易被人准确辨认出来。但如果考虑用计算机来识别数字，就不一定完全遵循数字的直观显示了，可以有一种内部的映射。若去掉某一段数码管后不影响辨认这 10 个数字，则得出的新映射关系就可以理解成原始 7 段数码管形式的一个约简。利用下面的 MATLAB 语句可以立即得出下面的约简结果，若最后一个语句调用 `redu()` 函数也将得到同样的结果。

表 10-14 数码管显示真值表

数码	C							D	数码	C							D
X	a	b	c	d	e	f	g	值	X	a	b	c	d	e	f	g	值
0	1	1	1	1	1	1	0	0	5	1	0	1	1	0	1	1	5
1	0	1	1	0	0	0	0	1	6	1	0	1	1	1	1	1	6
2	1	1	0	1	1	0	1	2	7	1	1	1	0	0	0	0	7
3	1	1	1	1	0	0	1	3	8	1	1	1	1	1	1	1	8
4	0	1	1	0	0	1	1	4	9	1	1	1	1	0	1	1	9

```
>> C=[1,1,1,1,1,1,0; 0,1,1,0,0,0,0; 1,1,0,1,1,0,1; 1,1,1,1,0,0,1;
      0,1,1,0,0,1,1; 1,0,1,1,0,1,1; 1,0,1,1,1,1,1; 1,1,1,0,0,0,0;
      1,1,1,1,1,1,1; 1,1,1,1,0,1,1];
   D=[0; 1; 2; 3; 4; 5; 6; 7; 8; 9]; X=[C D];
   c=1:7; d=8; Y=core(c,d,X)  % 其中 1-7 列为 C 属性, 8 为 D 属性
Y =
     1     2     5     6     7
```

可见，1,2,5,6,7 段数码管是不能约简掉的，而 3,4 (即图 10-41 (a) 中的 c,d 段) 是可有可无的，去掉它们并不影响辨认数码管显示的数字，可以用图 10-41 (b) 中的映射关系辨认数字。这样做虽然对人工辨认没有什么好处，但对机器辨认数字无疑会很方便，为机器视觉和书写体数字识别等领域是很有用途的。

【例 10-30】SARS 是 2003 年给全球带来恐慌的疾病，其准确诊断是很困难的。这里给出从报刊提取出的一些数据，构成表 10-15，试利用粗糙集理论对给出的 12 个条件进行约简，找出辅助诊断的最主要的条件。**这里的数据有些不确切，数据样本也不完全，所以不能真正用于临床诊断。**

表 10-15 SARS 患者和正常人若干检测指标表

	C 属性												D
U	干咳	呼吸困难	血液检测	高烧 38°C	X 射线	浓痰	白细胞多	寒战	肌肉酸痛	乏力	胸膜痛	头痛	SARS
1	1	1	1	1	0	0	0	0	1	1	0	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	1	0	0	0	0	0	0	1	0	0	0
4	0	0	0	1	1	1	1	0	1	0	1	1	0
5	1	0	0	1	1	1	1	1	0	1	1	0	0
6	0	1	0	1	1	1	1	1	1	0	0	1	0
7	1	0	0	0	1	1	1	0	0	1	1	1	0
8	1	1	1	1	0	0	0	0	1	1	0	1	1
9	1	0	1	1	1	0	0	0	1	1	0	1	1
10	1	1	1	1	0	0	0	0	1	1	0	1	1
11	1	0	1	1	1	0	0	0	1	1	0	1	1
12	1	0	1	1	1	0	0	0	1	1	0	1	1

【求解】 根据题意，可以给出如下的命令来进行条件约简，最后得出的条件为 3,4，表示第 3 和 4 列是诊断 SARS 的重要因素，亦即“血液检测呈阳性”和“高烧 38°C”。

```
>> D=[1; 0; 0; 0; 0; 0; 0; 1; 1; 1; 1; 1];
   C=[1,1,1,1,0,0,0,0,1,1,0,1; 0,0,0,0,0,0,0,0,0,0,0,0;
      1,0,1,0,0,0,0,0,0,1,0,0; 0,0,0,1,1,1,1,0,1,0,1,1;
      1,0,0,1,1,1,1,1,0,1,1,0; 0,1,0,1,1,1,1,1,1,0,0,1;
      1,0,0,0,1,1,1,0,0,1,1,1; 1,1,1,1,0,0,0,0,1,1,0,1;
      1,0,1,1,1,0,0,0,1,1,0,1; 1,1,1,1,0,0,0,0,1,1,0,1;
      1,0,1,1,1,0,0,0,1,1,0,1; 1,0,1,1,1,0,0,0,1,1,0,1];
   Y=redu(1:12,13,[C D])
 Y =
     3     4
```

10.5.2.3 粗糙集约简的 MATLAB 程序界面

基于粗糙集约简的理论和方法，还编写了 MATLAB 程序界面。在 MATLAB 提示符下键入 rsdav3 命令则将启动该程序界面，得出如图 10-42 所示的对话框，用户可以由其中的 Browse 按钮读入信息系统决策表，给出 C 属性和 D 属性所需的列号，则可以进一步进行分析。例如，单击 Redu 按钮则可以进行约简，结果将在 Results 栏目显示出来。

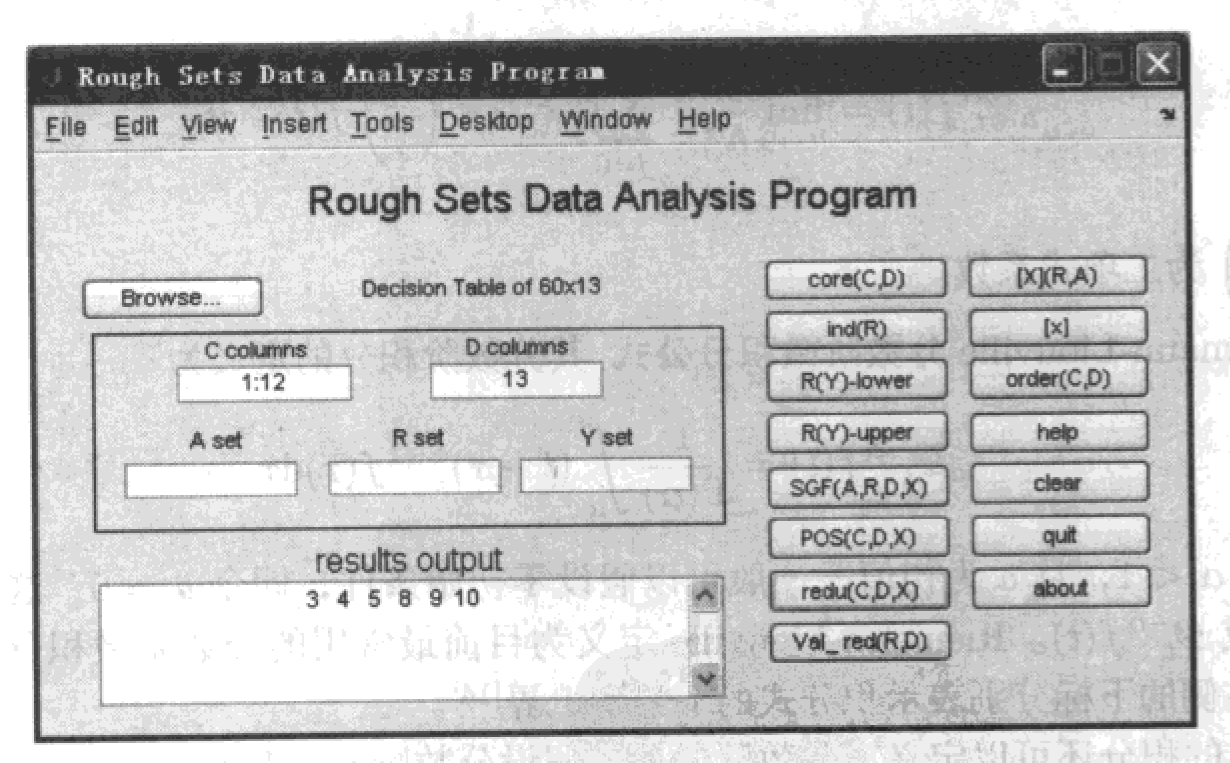


图 10-42 粗糙集数据分析程序界面

10.6 分数阶微积分学及其应用

第 3 章详细介绍了微积分学的相关内容，其他一些章节还陆续介绍了基于 MATLAB 语言的微积分问题的计算方法。一般地，$\mathrm{d}^n y/\mathrm{d}x^n$ 表示 y 对 x 的 n 阶导数，但若 $n=1/2$ 时是什么含义呢？这是 300 多年以前法国著名数学家 Guillaume François Antoine

L'Hôpital 问过微积分学创造者之一 Gattfried Wilhelm Leibnitz 的一个问题[47]。分数阶微积分理论建立至今已经有 300 年的历史了，但早期主要侧重于理论研究，近年来在很多领域都已经开始应用分数阶微积分学理论，例如在自动控制领域出现了分数阶控制理论等新的分支。本节将介绍分数阶微积分的定义及各种计算方法，并介绍分数阶线性及非线性微分方程的求解方法。

10.6.1 分数阶微积分的定义

10.6.1.1 分数阶微积分的各种定义及性质

在分数阶微积分理论发展过程中，出现了很多种函数的分数阶微积分的定义，如由整数阶微积分直接扩展而来的 Cauchy 积分公式、Grünwald-Letnikov 分数阶微积分定义、Riemann-Liouville 分数阶微积分定义以及 Capotu 定义等，本节将先介绍这些定义及其等效关系，再给出分数阶微积分的各种性质：

① **分数阶 Cauchy 积分公式** 该公式从简单整数阶积分直接扩展而来。

$$\mathscr{D}^{\gamma} f(t)=\frac{\Gamma(\gamma+1)}{2\pi \mathrm{j}} \int_{\mathrm{C}} \frac{f(\tau)}{(\tau-t)^{\gamma+1}} \mathrm{d}\tau \tag{10-6-1}$$

其中，C 为包围 $f(t)$ 单值与解析开区域的光滑曲线。

② Grünwald-Letnikov **分数阶微积分定义** 该定义为

$${}_a\mathscr{D}_t^{\alpha} f(t)=\lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{[(t-a)/h]}(-1)^j\binom{\alpha}{j} f(t-jh) \tag{10-6-2}$$

其中，$\binom{\alpha}{j}$ 为二项式系数。

③ Riemann-Liouville **分数阶微积分公式** 其分数阶积分的定义为

$${}_a\mathscr{D}_t^{-\alpha} f(t)=\frac{1}{\Gamma(\alpha)} \int_a^t(t-\tau)^{\alpha-1} f(\tau) \mathrm{d}\tau \tag{10-6-3}$$

其中，$0<\alpha<1$，且 a 为初值，一般可以假设零初始条件，即令 $a=0$，这时微分记号可以简写成 $\mathscr{D}_t^{-\alpha} f(t)$。Riemann-Liouville 定义为目前最常用的分数阶微积分定义。特别地，$\mathscr{D}$ 左右侧的下标分别表示积分式的下界和上界[14]。

由这样的积分还可以定义出分数阶微分。假设分数阶 $n-1<\beta \leqslant n$，则定义其分数阶微分为

$${}_a\mathscr{D}_t^{\beta} f(t)=\frac{\mathrm{d}^n}{\mathrm{d}t^n}\left[{}_a\mathscr{D}_t^{-(n-\beta)} f(t)\right]=\frac{1}{\Gamma(n-\beta)} \frac{\mathrm{d}^n}{\mathrm{d}t^n}\left[\int_a^t \frac{f(\tau)}{(t-\tau)^{\beta-n+1}} \mathrm{d}\tau\right] \tag{10-6-4}$$

④ Caputo **分数阶微分定义** Caputo 分数阶微分定义为

$${}_0\mathscr{D}_t^{\alpha} y(t)=\frac{1}{\Gamma(1-\gamma)} \int_0^t \frac{y^{(m+1)}(\tau)}{(t-\tau)^{\gamma}} \mathrm{d}\tau \tag{10-6-5}$$

其中，$\alpha = m + \gamma$, m 为整数，$0 < \gamma \leqslant 1$。类似地，Caputo 分数阶积分定义为

$$ {}_0\mathscr{D}_t^{\gamma} = \frac{1}{\Gamma(-\gamma)} \int_0^t \frac{y(\tau)}{(t-\tau)^{1+\gamma}} \mathrm{d}\tau, \ \ \gamma < 0 \tag{10-6-6}$$

可以证明[38]，对很广一类实际函数来说，Grünwald-Letnikov 分数阶微积分定义及 Riemann-Liouville 分数阶微积分定义是完全等效的。Caputo 定义和 Riemann-Liouville 定义的区别主要表现在对常数求导的定义上，前者对常数的求导是有界的 (为 0)，而后者求导是无界的，Caputo 定义更适用于分数阶微分方程初值问题的描述。本节主要研究 Riemann-Liouville 分数阶微积分问题，Cauchy 积分定义和其他几种定义有一些区别，这将在后面内容中通过例子来演示。

分数阶微积分有如下性质[37]：

① 解析函数 $f(t)$ 的分数阶导数 ${}_0\mathscr{D}_t^{\alpha} f(t)$ 对 t 和 α 都是解析的。

② $\alpha = n$ 为整数时，分数阶微分与整数阶微分的值完全一致，且 ${}_0\mathscr{D}_t^{0} f(t) = f(t)$。

③ 分数阶微积分算子为线性的，即对任意常数 a, b，有

$$ {}_0\mathscr{D}_t^{\alpha} \left[af(t) + bg(t) \right] = a\, {}_0\mathscr{D}_t^{\alpha} f(t) + b\, {}_0\mathscr{D}_t^{\alpha} g(t) \tag{10-6-7}$$

④ 分数阶微积分算子满足交换律，并满足叠加关系

$$ {}_0\mathscr{D}_t^{\alpha} \left[{}_0\mathscr{D}_t^{\beta} f(t) \right] = {}_0\mathscr{D}_t^{\beta} \left[{}_0\mathscr{D}_t^{\alpha} f(t) \right] = {}_0\mathscr{D}_t^{\alpha+\beta} f(t) \tag{10-6-8}$$

10.6.1.2 分数阶微积分的积分变换

函数的分数阶积分表达式的 Laplace 变换为

$$ \mathscr{L}\left[\mathscr{D}_t^{-\gamma} f(t) \right] = s^{-\gamma} \mathscr{L}[f(t)] \tag{10-6-9}$$

函数分数阶微分的 Laplace 变换为

$$ \mathscr{L}\left[{}_0\mathscr{D}_t^{\alpha} f(t) \right] = s^{\alpha} \mathscr{L}[f(t)] - \sum_{k=1}^{n-1} s^k \left[{}_0\mathscr{D}_t^{\alpha-k-1} f(t) \right]_{t=0} \tag{10-6-10}$$

特别地，若函数 $f(t)$ 及其各阶导数的初值均为 0，则 $\mathscr{L}[{}_0\mathscr{D}_t^{\alpha} f(t)] = s^{\alpha} \mathscr{L}[f(t)]$。

函数 $f(t)$ 的分数阶微积分的 Fourier 变换可以统一写成

$$ \mathscr{F}\left[{}_{-\infty}\mathscr{D}_t^{\alpha} f(t) \right] = (\mathrm{j}\omega)^{\alpha} \mathscr{F}[f(t)] \tag{10-6-11}$$

其中，α 既可以为正数，表示分数阶微分，也可以为负值，表示积分。

10.6.2 分数阶微积分的计算

10.6.2.1 利用 Fourier 级数计算周期函数的分数阶微积分

在介绍基于 Fourier 级数的分数阶微积分计算之前，有必要先回顾一下 Fourier 级数及其求解方法。对周期为 $2L$ 的函数，可以利用第 3.2.2 节介绍的 Fourier 展开方法，将

$t \in [-L, L]$ 区间内的信号 $f(t)$ 写成三角函数的级数形式，即

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos \frac{n\pi}{L} t + b_n \sin \frac{n\pi}{L} t \right) \tag{10-6-12}$$

其中，

$$\begin{cases} a_n = \dfrac{1}{L} \displaystyle\int_{-L}^{L} f(t) \cos \frac{n\pi t}{L} \mathrm{d}t, & n = 0, 1, 2, \cdots \\ b_n = \dfrac{1}{L} \displaystyle\int_{-L}^{L} f(t) \sin \frac{n\pi t}{L} \mathrm{d}t, & n = 1, 2, 3, \cdots \end{cases} \tag{10-6-13}$$

若 $t \in (a, b)$，则可以计算出 $L = (b-a)/2$。引入新变量 $\hat{t}$，使得 $t = \hat{t} + L + a$，则可以将 $f(\hat{t})$ 映射成 $(-L, L)$ 区间上的函数，可以对之进行 Fourier 级数展开，再将 $\hat{y} = t - L - a$ 转换成 t 的函数即可。用该节编写的函数 `fseries()` 可以得出 Fourier 级数及 Fourier 系数 a_n, b_n。

假设已知正弦、余弦函数，其整数阶微分表达式为

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k}\left[\sin at\right] = a^k \sin\left(at + \frac{k\pi}{2}\right), \quad \frac{\mathrm{d}^k}{\mathrm{d}t^k}\left[\cos at\right] = a^k \cos\left(at + \frac{k\pi}{2}\right) \tag{10-6-14}$$

由 Cauchy 积分公式可以证明，对分数阶的微分来说，当 k 为分数时，上述公式仍然成立[43]。所以，可以考虑用整数阶 Fourier 级数展开去逼近已知的 $f(t)$ 函数，利用式 (10-6-14) 给出的式子，就可以将 $f(t)$ 函数的 γ 阶导数用下面的公式直接计算出来。

$${}_a\mathscr{D}_t^{\gamma} f(t) = \frac{a_0}{\Gamma(1-\gamma)} t^{-\gamma} + \sum_{n=1}^{\infty} \left(\frac{n\pi}{L}\right)^{\gamma} \left[a_n \cos\left(\frac{n\pi}{L} t + \frac{\gamma\pi}{2}\right) + b_n \sin\left(\frac{n\pi}{L} t + \frac{\gamma\pi}{2}\right) \right] \tag{10-6-15}$$

若 γ 为大于 1 的分数，则可以将其分解成 $m + \gamma^*$。其中，m 为正整数，$0 < \gamma^* < 1$。这时可以先对原 $f(t)$ 函数先求取 m 阶导数，再对结果求 γ^* 阶微分。

若 γ 为负数，则立即计算出 $|\gamma|$ 阶积分的值。根据上述的算法，可以编写出基于 Fourier 级数展开的分数阶微分函数。其中，$0 < \gamma < 1$ 时建议采用此方法，否则用户可以自己决定是否需要将 γ 分解成整数与小数之和的形式再进行微积分运算。

```
function F=fdiffint(A,B,t,gam,a,b)
a0=A(1); A=A(2:end); n=length(B); L=(b-a)/2;
if gam>=0, F=0; else, F=a0*t^gam/gam; end
for i=1:n
    an=i*pi/L; bn=gam*pi/2;
    F=F+an^gam*(A(i)*cos(an*t+bn)+B(i)*sin(an*t+bn));
end
if a+b, F=subs(F,t,t-L-a); end
```

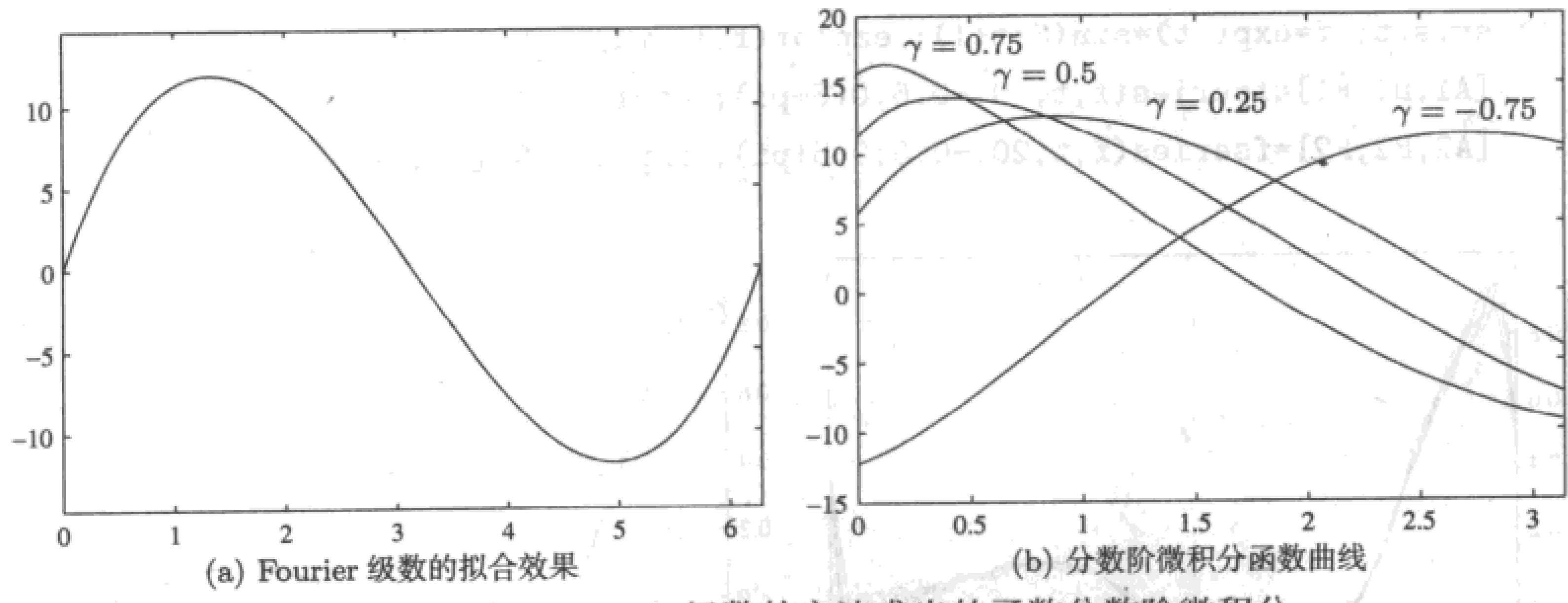
【例 10-31】 考虑例 3-20 中给出的函数 $y = t(t-\pi)(t-2\pi), t \in (0, 2\pi)$，试用 Fourier 级数展开的方法绘制出该函数的一些分数阶微分与积分函数曲线。

【求解】 利用 Fourier 级数展开方法，可以得出如图 10-43 (a) 所示的拟合效果。可见，对本例来说，直接采用 Fourier 级数，只用 12 项就能得出较好的拟合效果。

```
>> syms t; f=t*(t-pi)*(t-2*pi); [A,B,F]=fseries(f,t,12,0,2*pi);
   ezplot(f,[0,2*pi]); hold on; ezplot(F,[0,2*pi])
```

利用有限项的 Fourier 级数，就可以得出各个阶次下的微积分函数曲线，如图 10-43 (b) 所示。

```
>> F1=fdiffint(A,B,t,0.25,0,2*pi); ezplot(F1,[0,pi]); hold on
   F1=fdiffint(A,B,t,0.5,0,2*pi); ezplot(F1,[0,pi]);
   F1=fdiffint(A,B,t,0.75,0,2*pi); ezplot(F1,[0,pi]);
   F1=fdiffint(A,B,t,-0.75,0,2*pi); ezplot(F1,[0,pi]);
```



(a) Fourier 级数的拟合效果　　(b) 分数阶微积分函数曲线

图 10-43 基于 Fourier 级数的方法求出的函数分数阶微积分

用下面语句还可以绘制出分数阶微积分曲面，如图 10-44 所示。

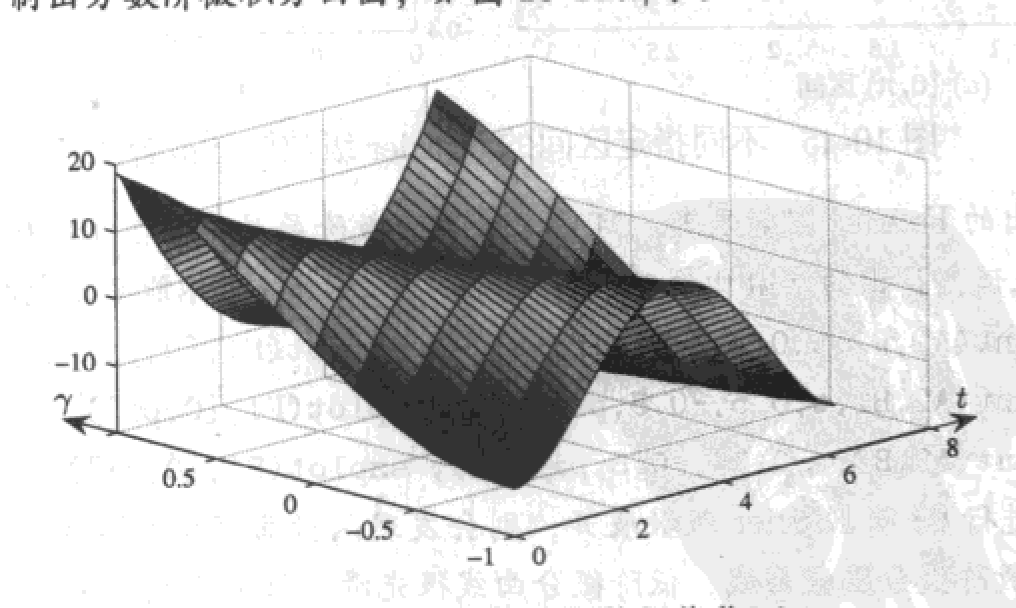


图 10-44 不同阶次下的微积分曲面

```
>> tt=0:0.1:2*pi; aa=-1:0.2:1; T=[];
   for r=aa
```

```
    F1=fdiffint(A,B,t,r,0,2*pi); T=[T; double(subs(F1,t,tt))];
end
hold off; surf(tt,aa,T)
```

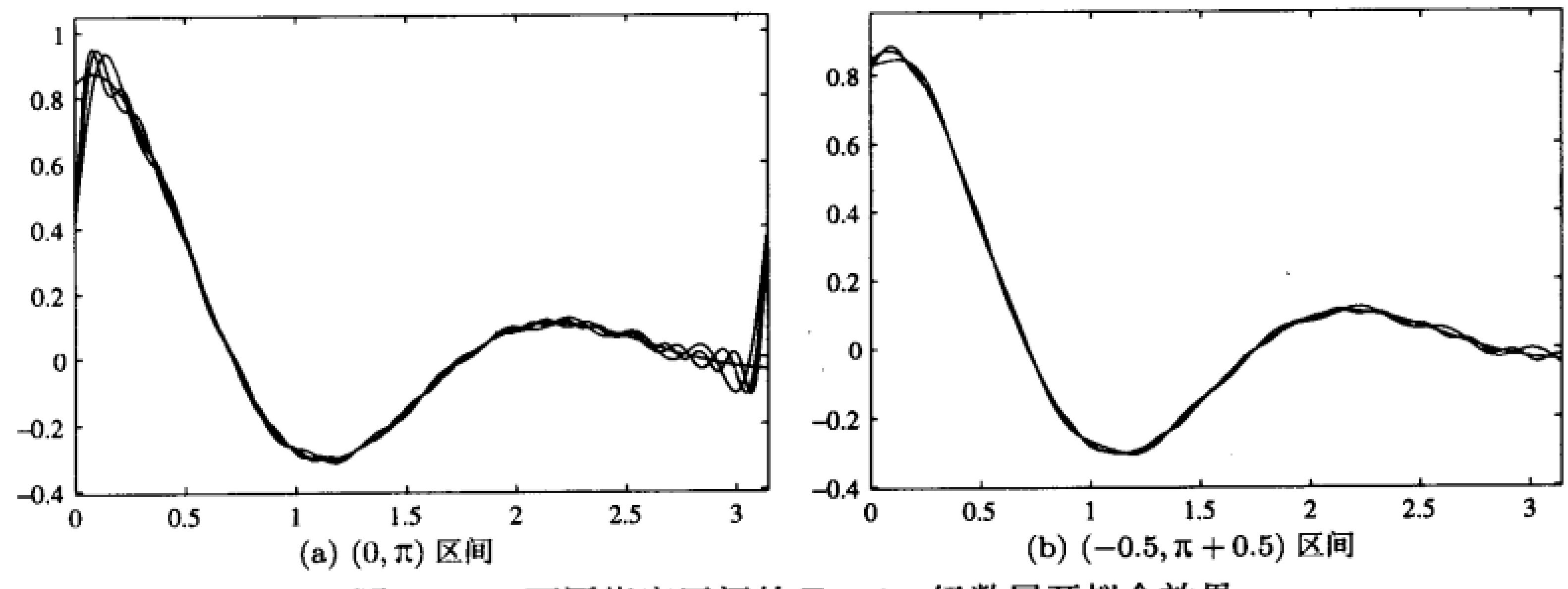
【例 10-32】假设给定函数 $f(t) = \mathrm{e}^{-t}\sin(3t+1)$，$t \in (0,\pi)$，试求出其分数阶导数。

【求解】对给定的区间进行 Fourier 级数展开，用不同的项数进行拟合，则将得出如图 10-45 (a) 所示的拟合效果。显然，这些拟合效果都不好，尤其表现在曲线的边界上。

```
>> syms t; f=exp(-t)*sin(3*t+1); ezplot(f,[0,pi]); hold on
   [A1,B1,F1]=fseries(f,t,10,0,pi); ezplot(F1,[0,pi]);
   [A2,B2,F2]=fseries(f,t,15,0,pi); ezplot(F2,[0,pi]);
   [A3,B3,F3]=fseries(f,t,20,0,pi); ezplot(F3,[0,pi]);
```

为了解决这样的问题，可以适当放大函数的定义域。在区间 $(-0.5,\pi+0.5)$ 上重新拟合原函数，这样在 $(0,\pi)$ 区间段内原函数的拟合效果如图 10-45 (b) 所示。显然，在这一区间内拟合效果明显改善了。

```
>> syms t; f=exp(-t)*sin(3*t+1); ezplot(f,[0,pi]); hold on
   [A1,B1,F1]=fseries(f,t,10,-0.5,0.5+pi); ezplot(F1,[0,pi]);
   [A2,B2,F2]=fseries(f,t,20,-0.5,0.5+pi); ezplot(F2,[0,pi]);
```



(a) $(0,\pi)$ 区间 (b) $(-0.5,\pi+0.5)$ 区间

图 10-45 不同指定区间的 Fourier 级数展开拟合效果

利用这样得出的 Fourier 级数展开，可以求出其分数阶导数，如图 10-46 (a) 所示。然而，对这样给出的函数来说，得出的导数曲线不平滑，因为 Fourier 级数不能平滑地拟合原函数。

```
>> F1=fdiffint(A2,B2,t,0.25,-0.5,pi+0.5); ezplot(F1,[0,pi]); hold on
   F1=fdiffint(A2,B2,t,0.5,-0.5,pi+0.5); ezplot(F1,[0,pi])
   F1=fdiffint(A2,B2,t,0.75,-0.5,pi+0.5); ezplot(F1,[0,pi])
```

对原始函数进行 60 项 Fourier 级数展开，则会发现函数曲线拟合效果有显著改善，图 10-46 (a) 中还给出了分数阶微分函数曲线，低阶微分曲线很光滑，但高阶微分曲线仍然不光滑。

```
>> [A1,B1,F1]=fseries(f,t,60,-0.5,0.5+pi);
   F1=fdiffint(A1,B1,t,0.25,-0.5,pi+0.5); ezplot(F1,[0,pi]); hold on
   F1=fdiffint(A1,B1,t,0.5,-0.5,pi+0.5); ezplot(F1,[0,pi])
```

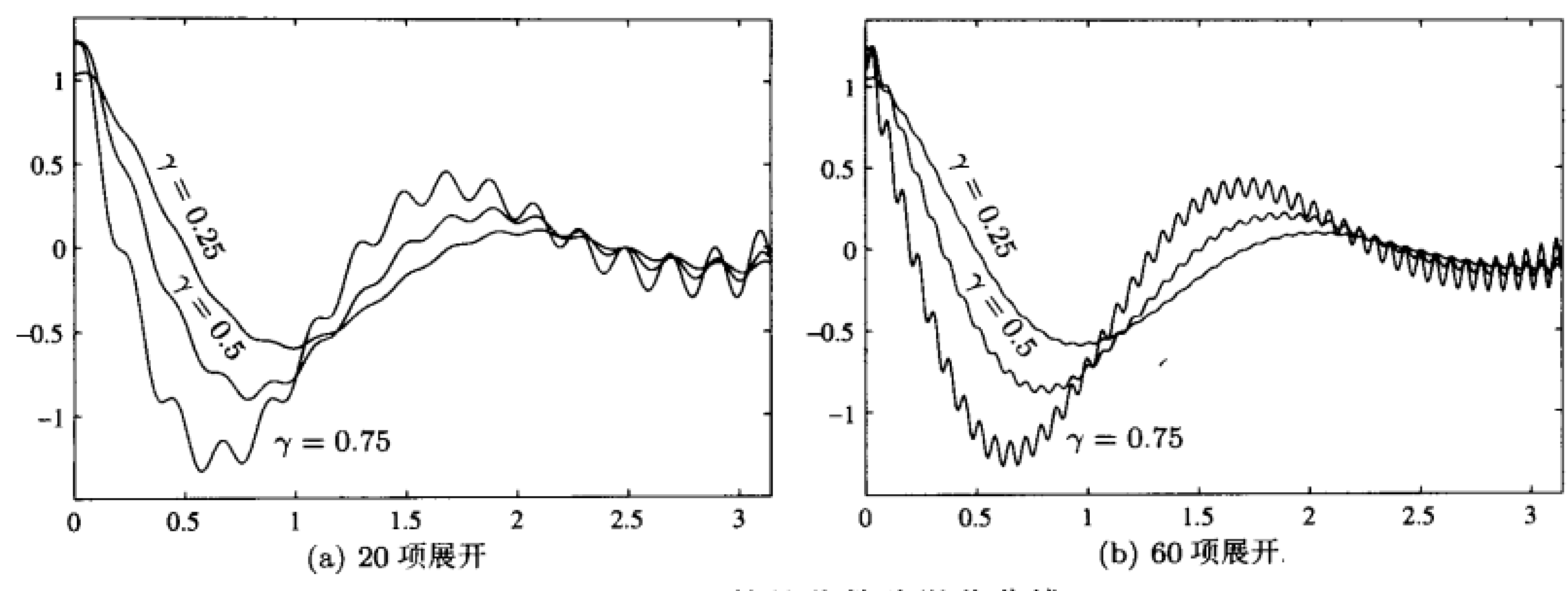


图 10-46 函数的分数阶微分曲线

```
F1=fdiffint(A1,B1,t,0.75,-0.5,pi+0.5); ezplot(F1,[0,pi])
```

基于 Fourier 级数的分数阶微积分计算方法完全取决于该函数是否能由 Fourier 级数精确近似，另外还需要已知该函数。

10.6.2.2 用 Grünwald-Letnikov 定义求解分数阶微分

其实，求解分数阶微积分的最直接数值方法是利用 Grünwald-Letnikov 定义的方法

$$ {}_a\mathscr{D}_t^\alpha f(t) = \lim_{h\to 0}\frac{1}{h^\alpha}\sum_{j=0}^{[(t-a)/h]}(-1)^j\binom{\alpha}{j}f(t-jh) \simeq \frac{1}{h^\alpha}\sum_{j=0}^{[(t-a)/h]}w_j^{(\alpha)}f(t-jh) \qquad (10\text{-}6\text{-}16)$$

其中，$w_j^{(\alpha)}=(-1)^j\binom{\alpha}{j}$ 为函数 $(1-z)^\alpha$ 的多项式系数，该系数还可以更简单地由下面的递推公式直接求出：

$$ w_0^{(\alpha)}=1,\ w_j^{(\alpha)}=\left(1-\frac{\alpha+1}{j}\right)w_{j-1}^{(\alpha)},\ j=1,2,\cdots \qquad (10\text{-}6\text{-}17)$$

假设步长 h 足够小，则可以用式 (10-6-16) 直接求出函数数值微分的近似值，并可以证明[38]，该公式的精度为 $o(h)$。所以，利用 Grünwald-Letnikov 定义可以立即编写出下面的函数来求取给定函数的分数阶微分函数。

```
function dy=glfdiff(y,t,gam)
h=t(2)-t(1); dy(1)=0; y=y(:); t=t(:);
w=1; for j=2:length(t), w(j)=w(j-1)*(1-(gam+1)/(j-1)); end
for i=2:length(t)
   dy(i)=w(1:i)*[y(i:-1:1)]/h^gam;
end
```

其中，$\boldsymbol{y},\boldsymbol{t}$ 分别为给定函数的采样值与时刻值构成的向量。要求 $\boldsymbol{t}$ 为等间距向量，且 gam 为分数阶的阶次，这样得出的 $\boldsymbol{d_y}$ 向量为函数的分数阶导数。

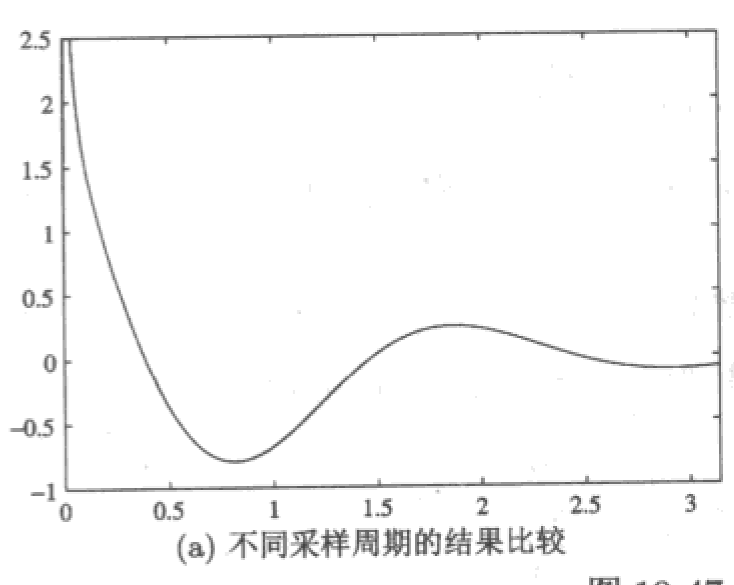
【例 10-33】考虑例 10-32 中给出的函数 $f(t)=\mathrm{e}^{-t}\sin(3t+1)$，$t\in(0,\pi)$，试求出其分数阶导数。

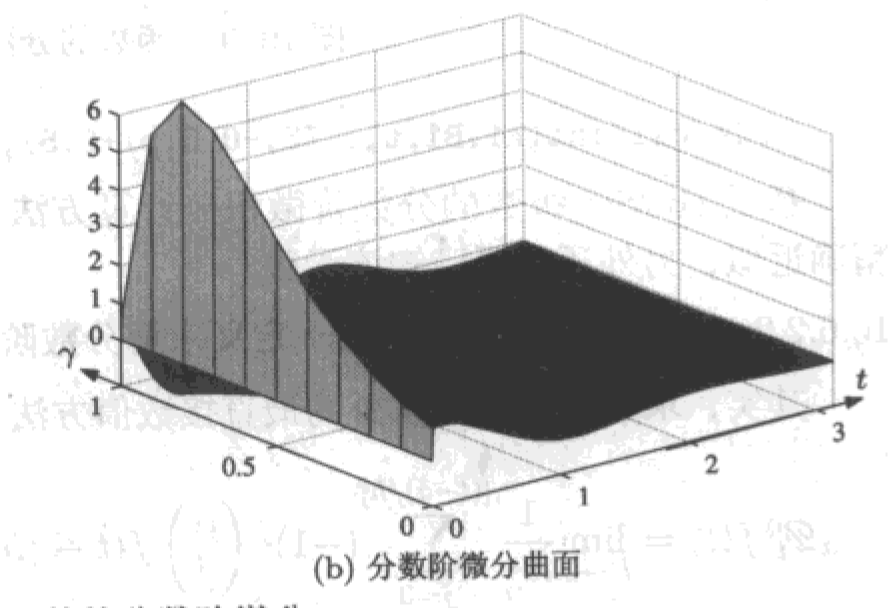
【求解】分别选择采样周期 $T=0.01$ 和 0.001，则可以得出在这两个采样周期下计算出的 0.5 阶导数函数曲线，如图 10-47 (a) 所示。可见，在 t 接近 0 的区域外二者是很接近的。对本函数而言，选择 $T=0.01$ 可以足够精确地求出函数的分数阶微分。

```
>> t=0:0.001:pi; y=exp(-t).*sin(3*t+1); dy=glfdiff(y,t,0.5); plot(t,dy);
   t=0:0.01:pi; y=exp(-t).*sin(3*t+1); dy=glfdiff(y,t,0.5); line(t,dy)
```

对不同的 γ 选值，可以调用下面的语句绘制出分数阶导函数的三维图，如图 10-47 (b) 所示。

```
>> Z=[]; t=0:0.01:pi; y=exp(-t).*sin(3*t+1);
   for gam=0:0.1:1, Z=[Z; glfdiff(y,t,gam)]; end
   surf(t,0:0.1:1,Z); axis([0,pi,0,1,-1.2,6])
```


(a) 不同采样周期的结果比较



(b) 分数阶微分曲面

图 10-47 函数的分数阶微分

【例 10-34】试用不同定义求取函数为 $f(t)=\sin(3t+1)$ 的 0.75 阶微分，并比较得出的结果。

【求解】由 Cauchy 定义，按式 (10-6-14) 给出的分数阶微分公式可以立即求出 0.75 阶微分为 ${}_0\mathscr{D}_t^{0.75}f(t)=3^{0.75}\sin\left(3t+1+\dfrac{0.75\pi}{2}\right)$，而用 Grünwald-Letnikov 定义的微分可以由 glfdiff() 函数得出。这样，由下面语句可以绘制出由这两个定义得出的分数阶微分曲线，如图 10-48 所示。

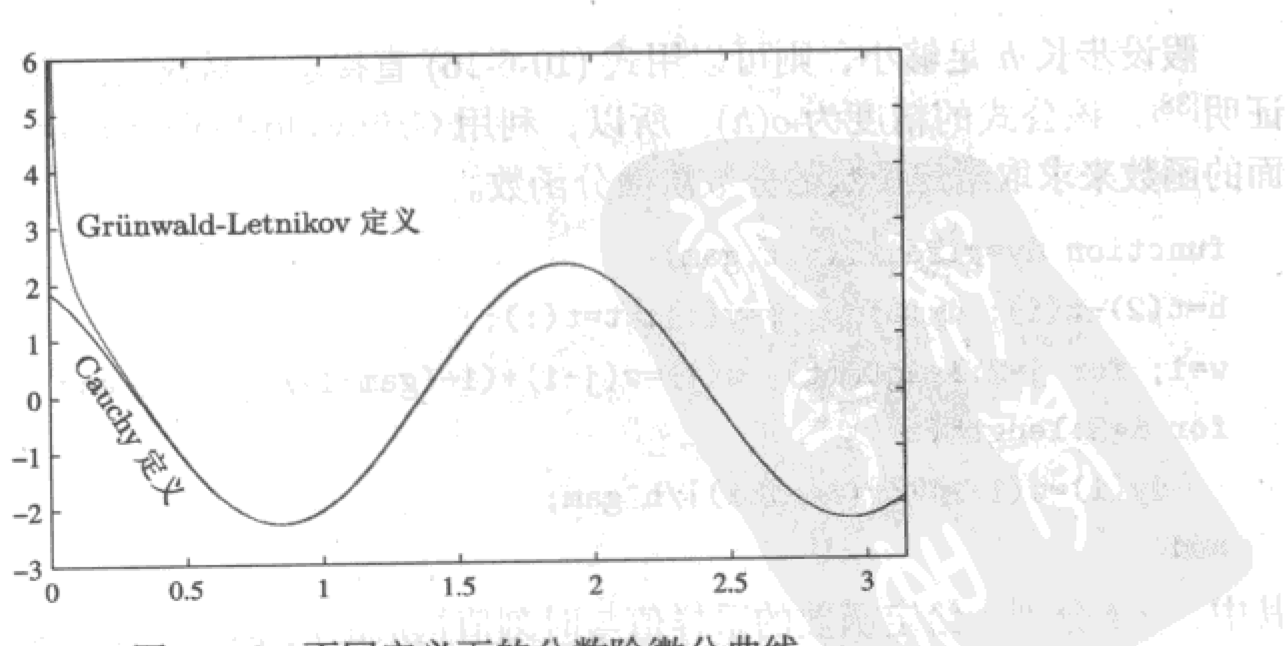


图 10-48 不同定义下的分数阶微分曲线

```
>> t=0:0.01:pi; y=sin(3*t+1); y1=3^0.75*sin(3*t+1+0.75*pi/2);
   y2=glfdiff(y,t,0.75); plot(t,y1,t,y2)
```

比较两种定义得出的结果，可见用 Cauchy 积分公式定义的算法没有初始突变过程，在 $t=0$ 区域以外二者几乎是一致的。从得出的结果很难断定哪个定义是正确的，因为这完全取决于 $t\leqslant 0$ 时 y 的值是什么。若 $y=0$，则显然在 $t=0^+$ 时刻 y 的值从 0 突变到 $\sin 1$，所以这时分数阶微分值应该为 ∞，突变的影响亦将持续一段时间，故 Grünwald-Letnikov 定义能正确反映该函数的分数阶微分变化，而不适合用 Cauchy 积分公式；若在 $t\leqslant 0$ 时原函数仍然满足函数 $y(t)=\sin(3t+1)$，则在 $t=0^+$ 时函数没有突变，则更适合使用 Cauchy 积分公式。

10.6.2.3 分数阶微积分的 Fourier 变换算法

由前面介绍的 Fourier 变换理论可以给定函数 $f(t)$ 的 Fourier 变换及反变换为

$$F(\omega)=\int_{-\infty}^{\infty}f(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t,\quad f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega \tag{10-6-18}$$

双边取分数阶微分，则可以得出

$$\begin{aligned}\mathscr{D}^{\gamma}f(t)&=\mathscr{D}^{\gamma}\left[\frac{1}{2\pi}\int_{-\infty}^{\infty}F(\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega\right]\\&=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(\omega)\mathscr{D}^{\gamma}\left[\mathrm{e}^{\mathrm{j}\omega t}\right]\mathrm{d}\omega=\frac{1}{2\pi}\int_{-\infty}^{\infty}(\mathrm{j}\omega)^{\gamma}F(\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega\end{aligned} \tag{10-6-19}$$

亦即该函数的 γ 阶微分实际上就是 $(\mathrm{j}\omega)^{\gamma}F(\omega)$ 函数的 Fourier 反变换。这些公式还有离散版本，对给定序列 $f(k)$，可以得出该序列的离散 Fourier 变换为

$$F(\omega)=\sum_{k=-\infty}^{\infty}f(k)\mathrm{e}^{-\mathrm{j}k\omega} \tag{10-6-20}$$

这样，函数分数阶导数可以表示成

$$\mathscr{D}^{\gamma}f(k)=\frac{1}{2\pi}\sum_{k=-\infty}^{\infty}(\mathrm{j}\omega)^{\gamma}F(\omega)\mathrm{e}^{\mathrm{j}k\omega} \tag{10-6-21}$$

10.6.2.4 分数阶微积分的滤波算法

前面介绍的各种分数阶微分运算的前题是被微分函数 $f(t)$ 为已知函数，但在实际应用中该信号经常是无法预先知道的，所以应该采用其他形式来求取分数阶微分，例如通过构造滤波器的方式来对信号进行数值微分处理。

信号的滤波器可以有连续和离散两种形式，分别用来拟合 Laplace 变换算子 s^{γ} 和 Fourier 变换算子 $(\mathrm{j}\omega)^{\gamma}$。从效果上看，函数的分数阶数值微分相当于原来信号需要通过这样的滤波器得出的输出信号。

文献 [37] 中列出了多种连续滤波器的实现算法。这里只介绍其中的 Oustaloup 算法[35]。假设选定的拟合频率段为 (ω_b,ω_h)，则可以构造出连续滤波器的传递函数模型为

$$G_f(s)=K\prod_{k=-N}^{N}\frac{s+\omega_k'}{s+\omega_k} \tag{10-6-22}$$

其中，滤波器零极点和增益可以由式 (10-6-23) 直接求出，为

$$\omega_k'=\omega_b\left(\frac{\omega_h}{\omega_b}\right)^{\frac{k+N+\frac{1}{2}(1-\gamma)}{2N+1}},\ \omega_k=\omega_b\left(\frac{\omega_h}{\omega_b}\right)^{\frac{k+N+\frac{1}{2}(1+\gamma)}{2N+1}},\ K=\left(\frac{\omega_h}{\omega_b}\right)^{-\frac{\gamma}{2}}\prod_{k=-N}^{N}\frac{\omega_k}{\omega_k'} \tag{10-6-23}$$

根据上述的算法，可以直接编写出如下的函数，设计连续滤波器。这样，若 $y(t)$ 信号通过滤波器进行过滤，则可以认为输出的信号是 $\mathscr{D}_t^{\gamma}y(t)$ 的近似。

```
function G=ousta_fod(r,N,w_L,w_H)
mu=w_H/w_L; k=-N:N;
w_kp=(mu).^((k+N+0.5-0.5*r)/(2*N+1) )*w_L;
w_k=(mu).^((k+N+0.5+0.5*r)/(2*N+1) )*w_L;
K=(mu)^(-r/2)*prod(w_k)/prod(w_kp);
G=tf(zpk(-w_kp',-w_k',K));
```

其中，r 为分数阶的阶次，$2N+1$ 为滤波器的阶次，w_L 和 w_H 分别为用户选定的拟合频率下限 ω_b 和上限 ω_h，一般在该区域内能较好地拟合分数阶微分算子，而其外的区域将和微分算子相差很多。由于算法本身可能的局限性，应该选择 $\omega_b\omega_h=1$。

【例 10-35】 假设 $\omega_b=0.01, \omega_h=100\,\mathrm{rad/sec}$，试设计出连续滤波器，对 $f(t)=\mathrm{e}^{-t}\sin(3t+1)$ 信号计算 0.5 阶微分。

【求解】 可以用下面的语句设计出

```
>> G=ousta_fod(0.5,2,0.01,100), bode(G)
Transfer function:
10 s^5 + 298.5 s^4 + 1218 s^3 + 768.5 s^2 + 74.97 s + 1
---------------------------------------------------------
 s^5 + 74.97 s^4 + 768.5 s^3 + 1218 s^2 + 298.5 s + 10
```

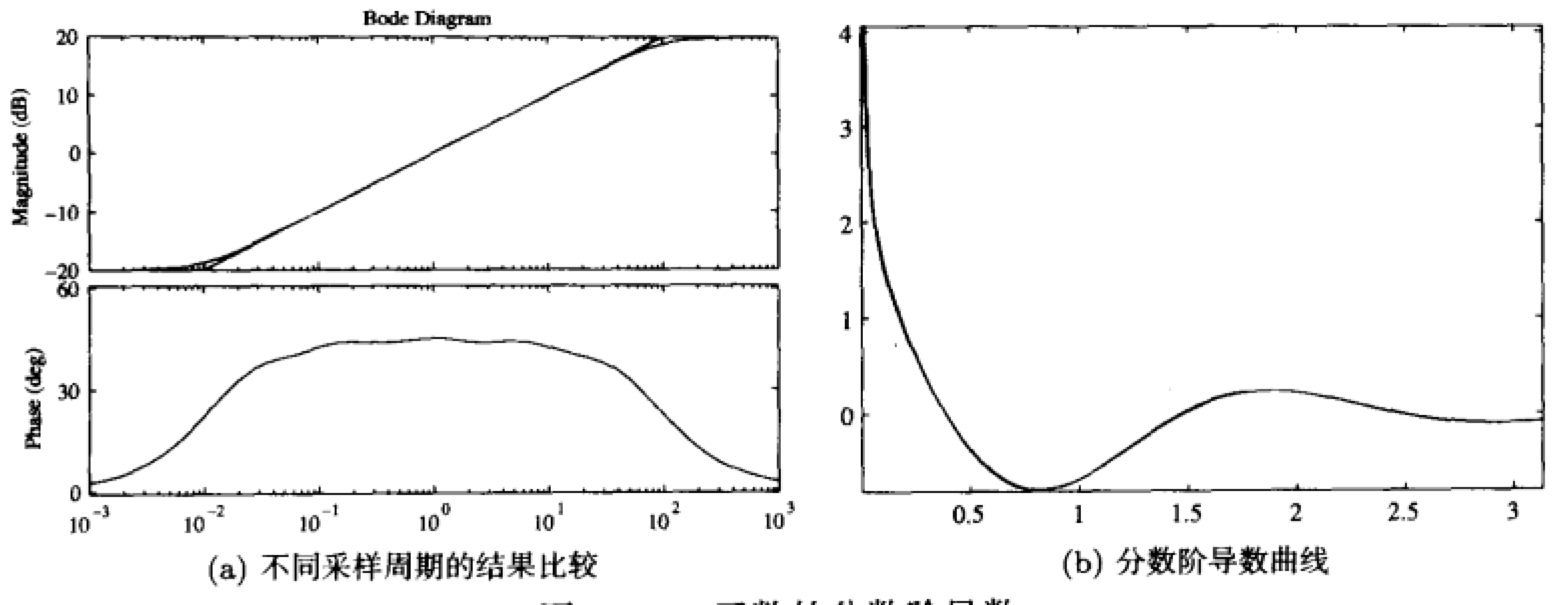
上面的语句还可以直接绘制出该滤波器的 Bode 图，如图 10-49 (a) 所示，在该图中还叠印了直线，表示 $(\mathrm{j}\omega)^{\gamma}$ 的理论值。由下面的语句还可以绘制出由滤波器计算出来的分数阶微分曲线，同时也将绘制出由 Grünwald-Letnikov 定义计算出来的分数阶微分曲线，如图 10-49 (b) 所示。可见，由滤波器计算出来的分数阶微分结果还是很精确的。

```
>> t=0:0.001:pi; y=exp(-t).*sin(3*t+1);
   y1=lsim(G,y,t); y2=glfdiff(y,t,0.5); plot(t,y1,t,y2)
```

当然，用该算法还可以在更大的频率范围内拟合分数阶微分函数，这时需要适当增大拟合的阶次。下面给出在 $(10^{-4},10^{4})$ 频段内的拟合效果，如图 10-50 所示。可见，对这样大的频率范围，不再适合 $N=2$ 的近似，而应该采用更大的 N 值，如 $N=4$。

```
>> G=ousta_fod(0.5,2,1e-4,1e4); G1=ousta_fod(0.5,3,1e-4,1e4);
   G2=ousta_fod(0.5,4,1e-4,1e4); G3=ousta_fod(0.5,5,1e-4,1e4);
   bode(G,'-',G1,'--',G2,':',G3,'-.')
```

对离散系统来说，由于纯粹的分数阶微分器是不能直接物理实现的，所以需要用 FIR 滤波器 [43]或 IIR 滤波器[4]的形式对其近似。利用滤波器设计工具箱中的 filt() 函数，由



(a) 不同采样周期的结果比较

(b) 分数阶导数曲线

图 10-49 函数的分数阶导数

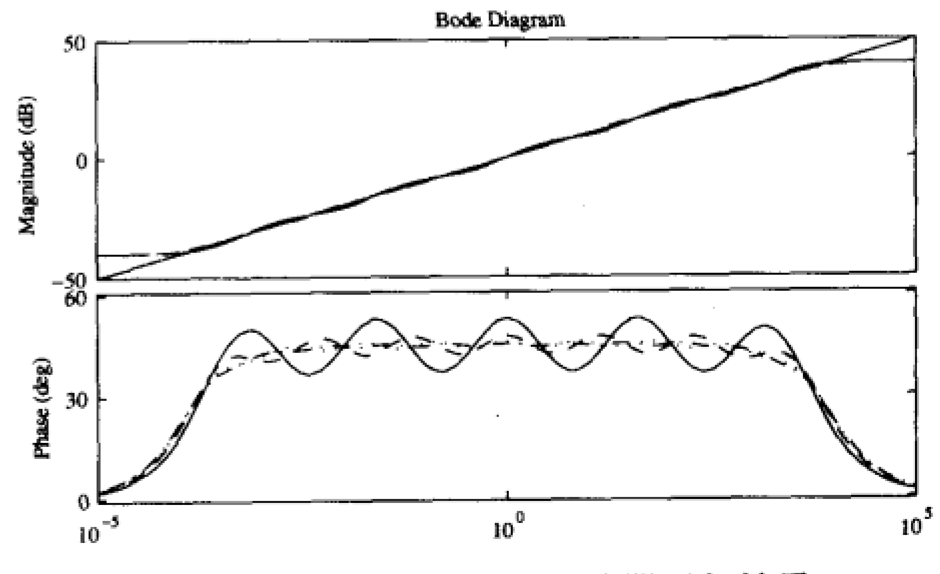


图 10-50 不同阶次下的滤波器近似效果

Ivo Petráš 博士开发的分数阶微积分器的 FIR 滤波器设计函数①。其核心部分清单如下：

```
function H=dfod2(n,T,r)
if r>0
   bc=cumprod([1,1-((r+1)./[1:n])]); H=filt(bc,[T^r],T);
elseif r<0
   bc=cumprod([1,1-((-r+1)./[1:n])]); H=filt([T^(-r)],bc,T);
end
```

其中，n 为期望的滤波器阶次，T 为滤波器的采样周期，r 为所需的导数阶次，用该函数可以直接设计出滤波器 H。若某信号经过滤波器 H，则可以得出该信号的 r 阶导数，经过滤波器得出的输出信号可以由 `filter()` 函数计算出来。

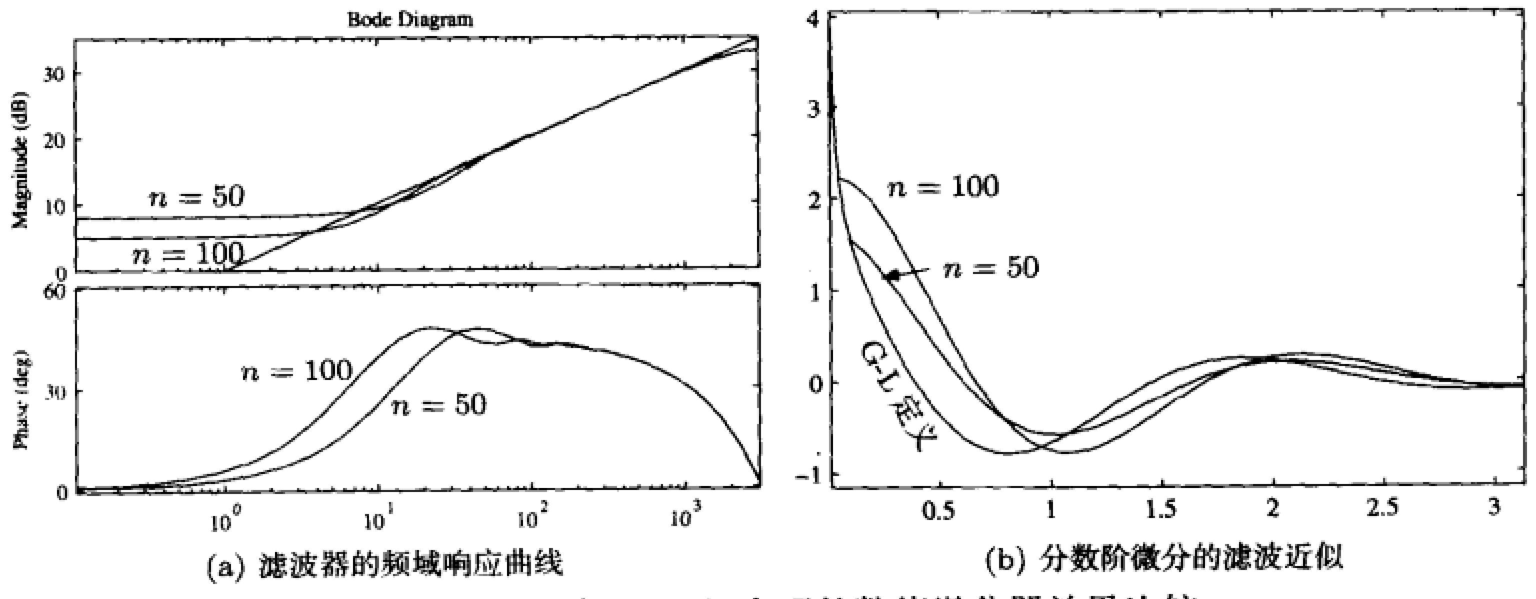
【例 10-36】 试用离散 FIR 滤波器对例 10-35 中给出的信号求取分数阶微分。

【求解】 用下面的语句可以直接设计出滤波器，并绘制出 Bode 图，如图 10-51 (a) 所示。由得出的频率响应看，$n = 100$ 明显优于 $n = 50$ 时的滤波器，但显然远不如前面介绍的用 Oustaloup 算

①可由 The MathWorks 免费下载，见 http://www.matlabcentral.com 下的信号处理滤波器设计栏目。

法设计的连续滤波器，在这些滤波器下还可以得出给定信号的分数阶微分，如图 10-51 (b) 所示，然而计算精度不甚理想。

```
>> t=0:0.001:pi; y=exp(-t).*sin(3*t+1); y3=glfdiff(y,t,0.5);
   G1=dfod2(50,0.001,0.5); G2=dfod2(100,0.001,0.5); bode(G1,G2)
   y1=filter(G1.num{1},G1.den{1},y); y2=filter(G2.num{1},G2.den{1},y);
   figure; plot(t,y1,t,y2,t,y3)
```



(a) 滤波器的频域响应曲线 (b) 分数阶微分的滤波近似

图 10-51 不同 (n,a) 组合下的数值微分器效果比较

FIR 滤波器的阶次较高，所以可以考虑采用 IIR 滤波器来生成分数阶微积分信号。从连续系统角度看，分数阶微积分可以用 Laplace 变换写成 $s^{\pm\gamma}$。对其进行离散化，则可以引入变换函数 $s=w\left(z^{-1}\right)$ 来近似分数阶微分或积分运算。

文献 [4] 给出了一种选择变换函数的新方法，该方法兼顾了 Simpson 积分算法和梯形积分算法，引入了加权系数 a，使得滤波器可以表示为 $H(z)=aH_S(z)+(1-a)H_T(s)$，$a\in[0,1]$。这样可以推导出 IIR 滤波器

$$G\left(z^{-1}\right)=k_0^{\pm\gamma}\left[\frac{1-z^{-2}}{(1+bz^{-1})^2}\right]^{\pm\gamma} \tag{10-6-24}$$

其中，$b=\dfrac{3+a-2\sqrt{3a}}{3-a}$，$k_0=\dfrac{6b}{T(3-a)}$，且 $\gamma\in(0,1)$，T 为离散化的采样周期。对高阶微积分来说，可以先进行整数阶微积分，对其结果再进行分数阶微积分。文献 [4] 还给出了基于连分式的滤波器设计方法和程序，对其修改后如下：

```
function H=iir_frac(r0,a,n,T)
syms x r b; aas=((1-x^2)/(1+b*x)^2)^r; n2=2*n; b0=(3+a-2*sqrt(3*a))/(3-a);
aas=char(subs(aas,{r,b},{r0,b0}));
maple('with(numtheory)'); maple(['cfe:=cfrac(' aas ',x,n2)']);
p=collect(maple('nthnumer','cfe',n2),x); n=sym2poly(p);
q=collect(maple('nthdenom','cfe',n2),x); d=sym2poly(q);
H=tf(n(end:-1:1)*(6*b0/T/(3-a))^r0,d(end:-1:1),'Variable','z^-1','Ts',T);
```

其中，r_0 为阶次，a 为加权系数，n 为滤波器阶次，T 为采样周期，这样可以由该函数直接设计出离散的滤波器模型 H。下面将通过例子演示 IIR 滤波器的设计及分数阶微积分近似解法。

【例 10-37】试用不同的 n,a 组合构造出 $s^{0.5}$ 阶滤波器，并比较其频域响应效果。

【求解】用前面给出的程序可以立即设计出如下的滤波器，并绘制出滤波器的 Bode 图，如图 10-52 (a) 所示。从得出的频率响应曲线看，若选择 $a=0.5$ 则相位曲线在给定的区域内不为恒值，所以拟合效果将不理想，但高频处赋值将有下降趋势，对噪声将有抑制作用。

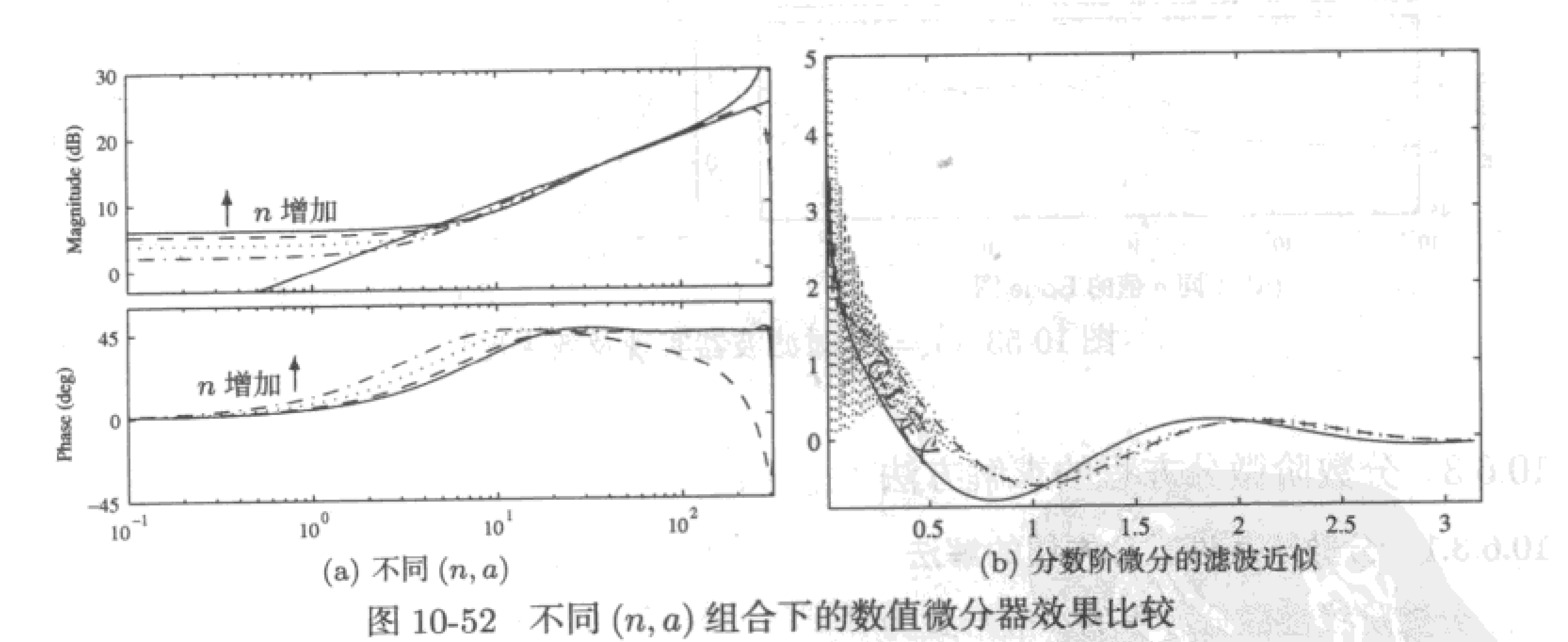
```
>> H1=iir_frac(0.5,0,3,0.01); H2=iir_frac(0.5,0.5,3,0.01);
   H3=iir_frac(0.5,0,4,0.01); H4=iir_frac(0.5,0,5,0.01);
   bode(H1,'-',H2,'--',H3,':',H4,'-.')
```

用 $H_{n,a}\left(z^{-1}\right)$ 来表示滤波器，则可以得出设计出的各类滤波器。例如，

$$H_{3,0}\left(z^{-1}\right)=\frac{113.1-56.57z^{-1}-56.57z^{-2}+14.14z^{-3}}{8+4z^{-1}-4z^{-2}-z^{-3}}$$

用下面的语句还可以绘制出各个滤波器的滤波效果，如图 10-52 (b) 所示。然而从得出的滤波效果看，阶次太低则近似效果不佳，所以在设计中应该适当增加拟合阶次。

```
>> t=0:0.01:pi; y=exp(-t).*sin(3*t+1); y0=glfdiff(y,t,0.5);
   y1=filter(H1.num{1},H1.den{1},y); y2=filter(H2.num{1},H2.den{1},y);
   y3=filter(H3.num{1},H3.den{1},y); y4=filter(H4.num{1},H4.den{1},y);
   plot(t,y0,t,y1,':',t,y2,'--',t,y3,':')
```



(a) 不同 (n,a)　　(b) 分数阶微分的滤波近似

图 10-52　不同 (n,a) 组合下的数值微分器效果比较

用 Maple 语言提供的连分式近似函数 cfrac() 有局限性，当选择的 $n\geqslant 6$ 不能得出有意义的近似，所以可以考虑 Padé 近似方法，用其直接拟合理想滤波器的 Taylor 幂级数展开式。这样，改写原来的离散化函数，可以写出如下的滤波器设计函数。该函数可以用于高阶滤波器的设计。

```
function H=iir_pade(r0,a,n,T)
syms x r b; aas=((1-x^2)/(1+b*x)^2)^r; b0=(3+a-2*sqrt(3*a))/(3-a);
```

```
aas=subs(aas,{r,b},{r0,b0}); c=taylor(aas,x,2*n);
c=sym2poly(c); c=c(end:-1:1); [n,d]=padefcn(c,n-1,n);
H=tf(n(end:-1:1)*(6*b0/T/(3-a))^r0,d(end:-1:1),'Variable','z^-1','Ts',T);
```

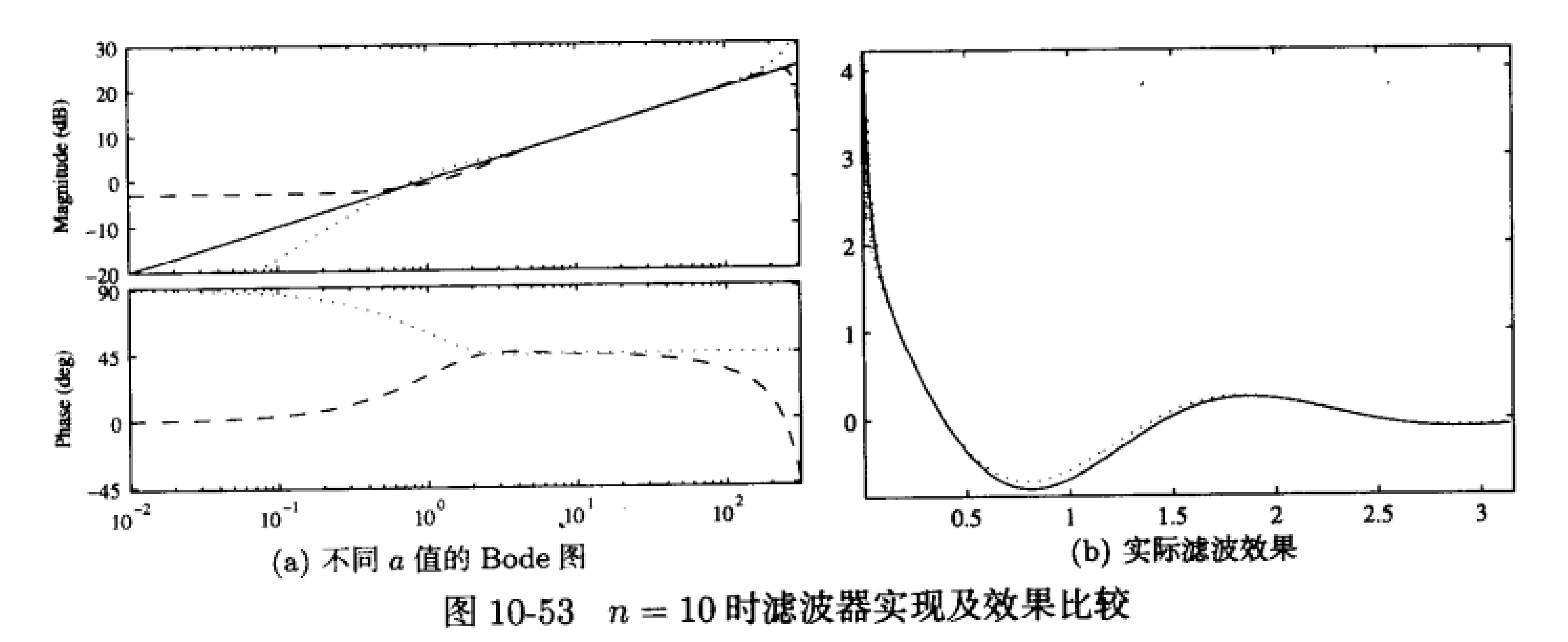
【例 10-38】 仍考虑例 10-36 中给出的函数，试提高滤波器的阶次，观察近似效果。

【求解】 采用 Maple 语言提供的连分式方法不能处理这样的高阶问题，所以应该采用上面给出的 Padé 算法，设计出相应的滤波器，并绘制出各个滤波器的 Bode 图，如图 10-53 (a) 所示。从实际滤波器的增益看，和真值相差较远，虽然比低价滤波器有所改善。

```
>> H1=iir_pade(0.5,0,10,0.01); H2=iir_pade(0.5,0.75,10,0.01);
   H2=minreal(H2); bode(H1,':',H2,'--')
```

用下面的语句还可以绘制出滤波效果，如图 10-53 (b) 所示。其中，$a = 0$ 的滤波器产生的分数阶微分信号被噪声淹没，所以这里只给出 $n = 10, a = 0.5$ 时滤波器的滤波效果。可见，采用高阶滤波器时精度将明显改善，较好地逼近了实际的分数阶信号。由此可以看出，IIR 滤波器比 FIR 滤波器表现出来的巨大优势。

```
>> t=0:0.01:pi; y=exp(-t).*sin(3*t+1); y0=glfdiff(y,t,0.5);
   y2=filter(H2.num{1},H2.den{1},y); plot(t,y0,t,y2,':')
```



(a) 不同 a 值的 Bode 图　　(b) 实际滤波效果

图 10-53　$n = 10$ 时滤波器实现及效果比较

10.6.3 分数阶微分方程的求解方法

10.6.3.1 分数阶线性微分方程的解法

分数阶线性微分方程的一般形式为[38]

$$a_n \mathscr{D}_t^{\beta_n} y(t) + a_{n-1} \mathscr{D}_t^{\beta_{n-1}} y(t) + \cdots + a_1 \mathscr{D}_t^{\beta_1} y(t) + a_0 \mathscr{D}_t^{\beta_0} y(t) = u(t) \tag{10-6-25}$$

其中，$u(t)$ 可以由某函数及其分数阶微分构成。为后面叙述方便，不妨对微分方程式作假设 $\beta_n > \beta_{n-1} > \cdots > \beta_1 > \beta_0 > 0$。若需要研究的微分方程出现下面两种特殊情况，则需要先对其变换，再进行求解。

① 若研究的微分方程各个阶次不满足上述大小关系，则可以先进行排序。

② 若涉及到负的 β_i 值，则原来的方程为分数阶微积分方程，需要选择引入新的变量 $z(t)=\mathscr{D}_t^{\beta_0}y(t)$，这样就可以将原来的微积分方程变换成关于 $z(t)$ 的微分方程了。

假设函数 $y(t)$ 具有零初始条件，则可以对该方程进行 Laplace 变换，得出

$$G(s)=\frac{Y(s)}{U(s)}=\frac{1}{a_n s^{\beta_n}+a_{n-1}s^{\beta_{n-1}}+\cdots+a_1 s^{\beta_1}+a_0 s^{\beta_0}} \tag{10-6-26}$$

这里，$G(s)$ 又称为分数阶传递函数。文献 [38] 中给出了求解分数阶线性微分方程的精确解法，然而该算法即使用计算机实现也有较大难度，所以这里讨论用其他数值算法。

考虑式 (10-6-16) 中给出的 Grüwald-Letnikov 定义，用离散方法可以将其改写成

$$ {}_a\mathscr{D}_t^{\beta_i}y(t)\simeq\frac{1}{h^{\beta_i}}\sum_{j=0}^{[(t-a)/h]}w_j^{(\beta_i)}y_{t-jh}=\frac{1}{h^{\beta_i}}\left[y_t+\sum_{j=1}^{[(t-a)/h]}w_j^{(\beta_i)}y_{t-jh}\right] \tag{10-6-27}$$

其中，$w_0^{(\beta_i)}$ 可以由下面的递推公式得出

$$w_0^{(\beta_i)}=1,\ \ w_j^{(\beta_i)}=\left(1-\frac{\beta_i+1}{j}\right)w_{j-1}^{(\beta_i)},\ j=1,2,\cdots \tag{10-6-28}$$

代入式 (10-6-25)，则可以直接推导出微分方程数值解为

$$y_t=\frac{1}{\displaystyle\sum_{i=0}^{n}\frac{a_i}{h^{\beta_i}}}\left[u_t-\sum_{i=0}^{n}\frac{a_i}{h^{\beta_i}}\sum_{j=1}^{[(t-a)/h]}w_j^{(\beta_i)}y_{t-jh}\right] \tag{10-6-29}$$

该算法可以直接由 MATLAB 实现。假设将各项的系数和微分阶次用 $\boldsymbol{a}$ 和 $\boldsymbol{n}$ 向量表示，并给出等间距时间向量 $\boldsymbol{t}$ 和这些点上的输入函数值 $\boldsymbol{u}$，则可以得出微分方程的数值解 $\boldsymbol{y}$ 向量。所以用该函数可以容易地求解分数阶线性微分方程。

```
function y=glfode(u,t,a,n)
h=t(2)-t(1); D=sum(a./[h.^n]); W=[]; y=zeros(length(t),1); w=y;
for i=1:length(n)
   w=1; for j=2:length(t), w(j)=w(j-1)*(1-(n(i)+1)/(j-1)); end
   W=[W; w];
end
for i=2:length(t)
   for j=1:length(n), A(j)=W(j,2:i)*[y(i-1:-1:1)]; end
   y(i)=(u(i)-sum(A.*a./[h.^n]))/D;
end
```

【例 10-39】 试用数值方法求解下面的分数阶线性微分方程。

$$\mathscr{D}_t^{3.5}y(t)+8\mathscr{D}_t^{3.1}y(t)+26\mathscr{D}_t^{2.3}y(t)+73\mathscr{D}_t^{1.2}y(t)+90\mathscr{D}_t^{0.5}y(t)=90\sin(t^2)$$

【求解】 由给出的方程可以写出 a 和 n 向量，从而直接调用编写的 glfode() 函数得出该微分方程的解，用绘图语句可以绘制出输出和输入信号的曲线，如图 10-54 所示。为提高得出数值解的精度，通常需要选择较小的 h 值，这里得出的结果精度较高，再进一步减小 h 的值，例如选择 $h = 0.001$，则得出的仿真结果与图中给出的结果看不出任何区别。

```
>> a=[1,8,26,73,90]; n=[3.5,3.1,2.3,1.2,0.5];
   t=0:0.002:10; u=90*sin(t.^2); y=glfode(u,t,a,n);
   subplot(211), plot(t,y); subplot(212), plot(t,u)
```

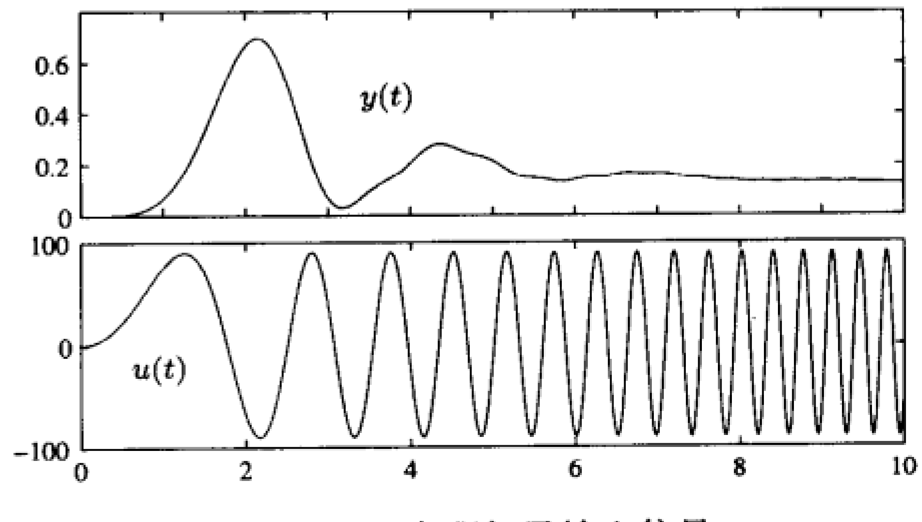


图 10-54 方程解及输入信号

在实际计算中对大规模问题有时计算量较大，所以可以考虑采用“短时记忆”效应的原理对求和式进行近似[38]。

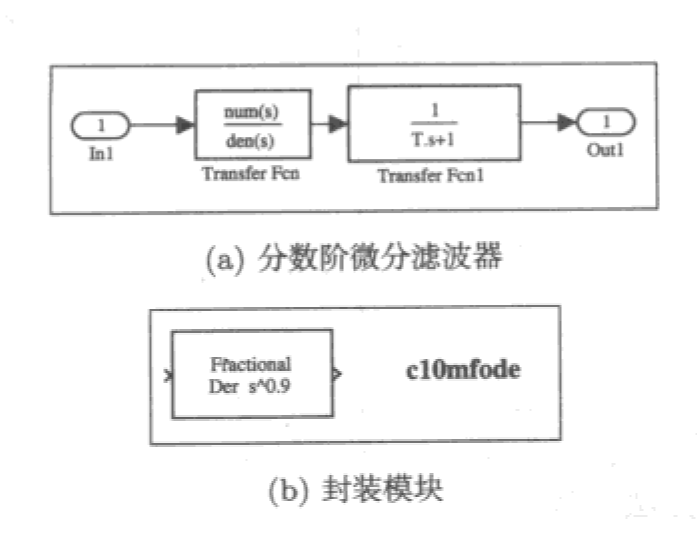
10.6.3.2 非线性分数阶微分方程近似解法

由前面的内容可见，对未知信号进行分数阶微分数值运算的一种有效途径是采用 Oustaloup 算法设计连续滤波器对信号进行滤波处理。另外，考虑到该滤波器分子和分母阶次一致，可能导致在仿真过程中出现代数环，所以应该在其后面再接一个低通滤波器，将其截止频率设置为 ω_h，这样可以建立起如图 10-55 (a) 所示的分数阶微分器模块，通过适当选择频段和阶次可以较好近似分数阶微分的效果。注意，虽然 Oustaloup 算法设计的滤波器理论上可以求取任意阶次的分数阶微积分，但从数值微积分精度看，该滤波器更适合求取 1 阶以内的分数阶微积分，所以应该将高阶微积分先进行整数阶微积分运算，再对结果进行滤波处理。

利用 Simulink 的模块封装技术[52]，可以将该模型进行封装，得出如图 10-55 (b) 所示的分数阶微分器模块。双击该模块则可以得出如图 10-55 (c) 所示的对话框，允许用户填写设计 Oustaloup 滤波器所需的参数。在模块封装初始化栏目应该填写下面的语句，以便在使用模块前先自动设计出滤波器，并根据阶次正确显示图标。

```
wb=ww(1); wh=ww(2); G=ousta_fod(gam,n,wb,wh);
num=G.num{1}; den=G.den{1};  T=1/wh; str='Fractional\n';
if isnumeric(gam)
   if gam>0, str=[str, 'Der  s^' num2str(gam) ];
   else, str=[str, 'Int  s^{' num2str(gam) '}']; end
```

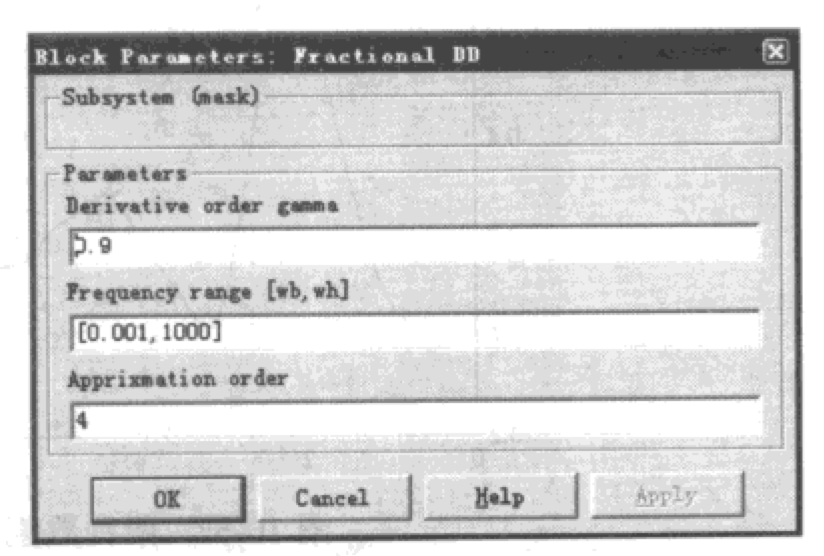
```
else, str=[str, 'Der  s^gam']; end
```



(a) 分数阶微分滤波器

Fractional Der s^0.9 c10mfode

(b) 封装模块



(c) 分数阶微分器参数对话框

图 10-55 分数阶数值微分设计模块

在实际仿真过程中，由于搭建起来的系统一般为刚性系统，所以在选择求解算法时应该选择 ode15s 或 ode23tb 等，因为这些算法可以保证较高的计算效率和精度。下面将通过例子演示该模块在分数阶微分方程近似求解中的应用。

【例 10-40】试用滤波器的思想求取例 10-39 中分数阶线性微分方程的数值解，并与该例中所用方法得出的结果进行比较。

【求解】求解分数阶线性微分方程问题不如例 10-39 中给出的方法直观。在求解之前，需要引入辅助变量 $z(t)=\mathscr{D}_t^{0.5}y(t)$，这样，原来的微分方程可以直接变换成下面的形式：

$$z(t)=\sin(t^2)-\frac{1}{90}\left[\mathscr{D}_t^3z(t)+8\mathscr{D}_t^{2.6}z(t)+26\mathscr{D}_t^{1.8}z(t)+73\mathscr{D}_t^{0.7}z(t)\right]$$

根据该方程可以搭建起如图 10-56 所示的 Simulink 仿真框图。对该框图进行仿真，则可以得出该微分方程的数值解。将两种方法得出的数值解在同一坐标系下绘制，则得出如图 10-57 所示的曲线。从曲线上基本看不出两者的差别，表明此算法可以以很高精度求解线性方程。

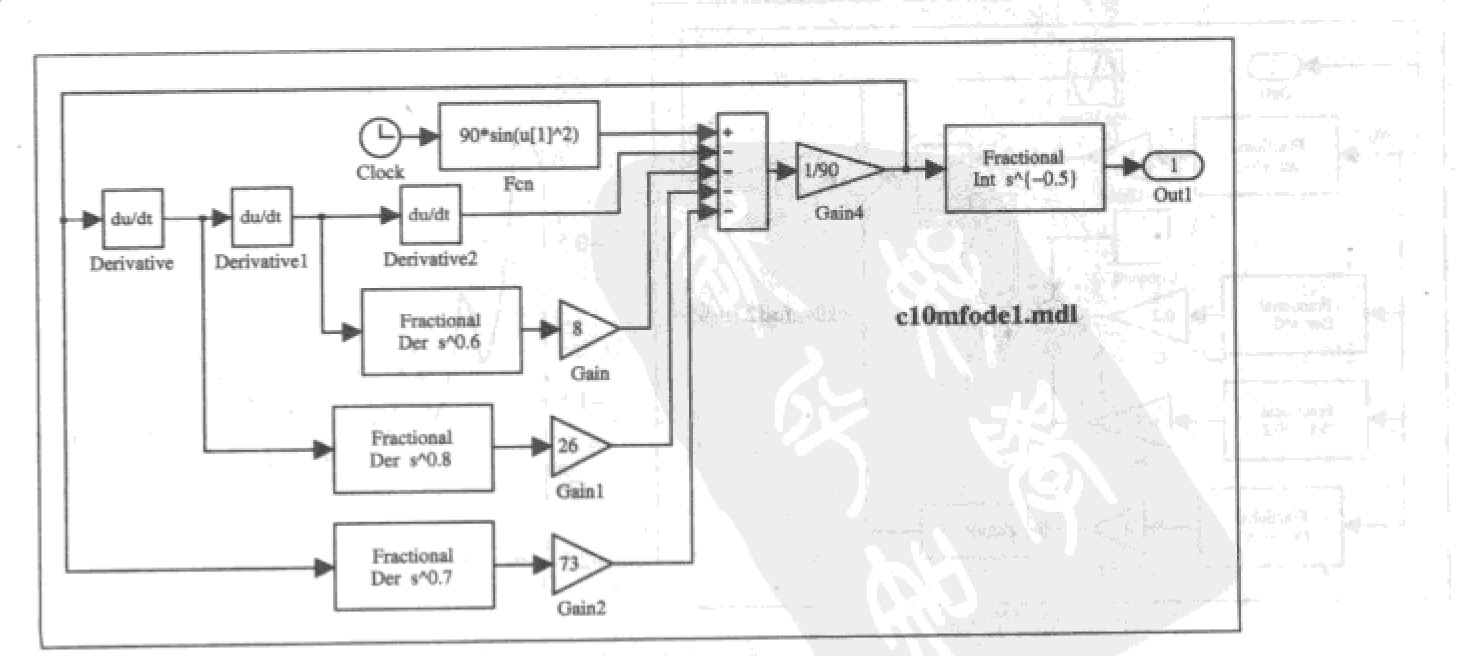


图 10-56 微分方程求解的 Simulink 框图

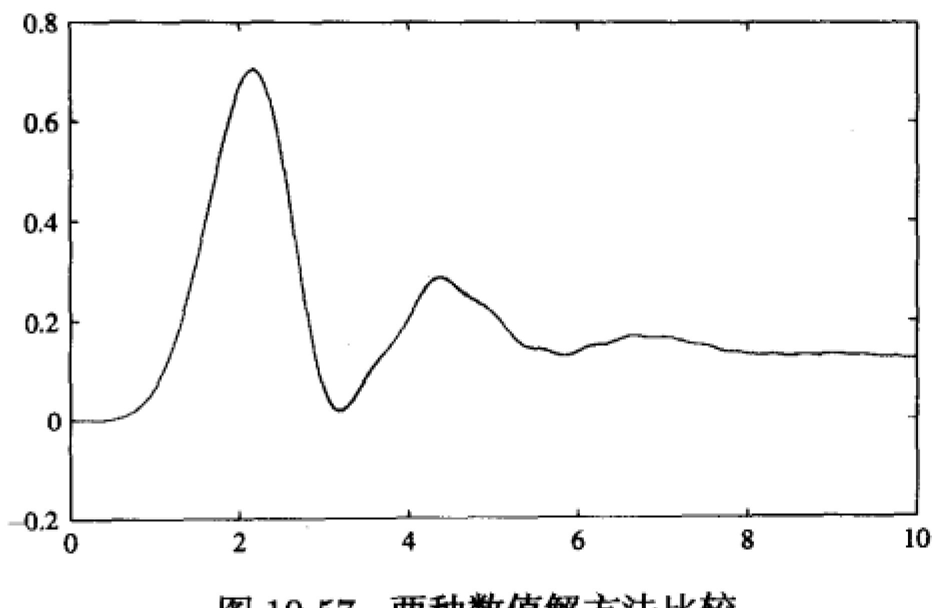


图 10-57 两种数值解方法比较

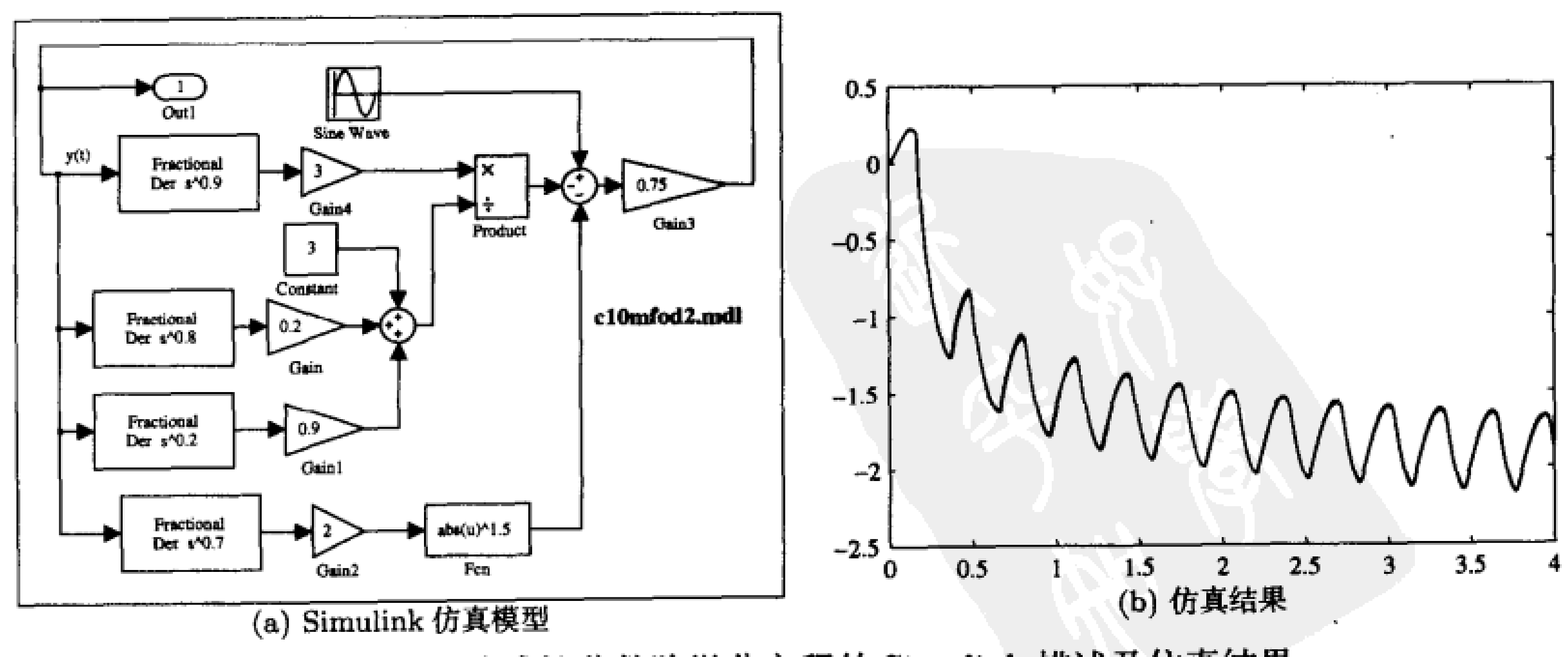
【例 10-41】 试用近似方法求解下面的分数阶非线性微分方程。

$$\frac{3\mathscr{D}^{0.9}y(t)}{3+0.2\mathscr{D}^{0.8}y(t)+0.9\mathscr{D}^{0.2}y(t)}+\left|2\mathscr{D}^{0.7}y(t)\right|^{1.5}+\frac{4}{3}y(t)=5\sin(10t)$$

【求解】 根据方程本身，可以容易地写出 $y(t)$ 函数的显式表达式为

$$y(t)=\frac{3}{4}\left[5\sin(10t)-\frac{3\mathscr{D}^{0.9}y(t)}{3+0.2\mathscr{D}^{0.8}y(t)+0.9\mathscr{D}^{0.2}y(t)}-\left|2\mathscr{D}^{0.7}y(t)\right|^{1.5}\right]$$

根据得出的 $y(t)$ 可以绘制出如图 10-58 (a) 所示的仿真模型。从得出的仿真模型可见，信号的各个分数阶微分信号可以由前面设计的模块获得，因此仿真的精度取决于滤波器对微分的拟合效果，选择不同的拟合频段和滤波器阶次对求解精度将有一定的影响。图 10-58 (b) 对不同的滤波器频段、阶次组合进行了比较，得出的结果基本一致，误差稍大的曲线是由 $\omega_b=0.001,\omega_h=1000,n=2$ 得出的。所以对此例来说，选择 $n=4$ 并选择适当的频段则得出的结果几乎完全一致。



(a) Simulink 仿真模型 (b) 仿真结果

图 10-58 非线性分数阶微分方程的 Simulink 描述及仿真结果

10.7 本章要点简介

- 本章介绍的各种方法的 MATLAB 函数由下表给出。

函数名	函 数 功 能	工具箱	本书页码
union()	集合的并运算	MATLAB	表 10-1
setdiff()	差集运算	MATLAB	表 10-1
intersect()	集合的交运算	MATLAB	表 10-1
setxor()	集合的异或运算	MATLAB	表 10-1
unique()	集合的惟一运算	MATLAB	表 10-1
ismember()	元素的属于判定	MATLAB	表 10-1
语句	集合的包含判定	自编	339
gbellmf()	钟形隶属函数计算	模糊逻辑	340
gaussmf()	Gauss 型隶属函数计算	模糊逻辑	341
mfedit()	隶属函数的图形界面调用	模糊逻辑	343
sigmf()	Sigmoid 型隶属函数计算	模糊逻辑	342
newfis()	建立模糊推理系统数据结构的函数	模糊逻辑	343
addvar()	给模糊推理系统添加输入输出变量的函数	模糊逻辑	344
fuzzy()	模糊推理系统设计程序界面	模糊逻辑	344
addrule()	向模糊推理系统的规则库补加新规则	模糊逻辑	346
evalfis()	已知模糊推理系统模型，求出给定输入下该系统输出的函数	模糊逻辑	347
newff()	前馈型神经网络结构对象建立	神经网络	349
属性设置	神经网络训练参数设定	神经网络	350
train()	神经网络训练函数	神经网络	350
plotperf()	神经网络训练中指标函数曲线绘制	神经网络	351
sim()	神经网络仿真函数，可以用于神经网络的泛化研究	神经网络	351
nntool()	神经网络研究用户界面	神经网络	356
gaopt()	基于遗传算法的最优化函数，第 364 页中给出了该函数更一般的调用格式，允许用户对其他参数进行选取，该函数是整个遗传算法最优化工具箱的用户接口函数	GAOT	360
ga()	遗传算法与直接搜索工具箱提供的最优化函数，该工具箱还提供了遗传算法参数设定的 gaoptimset() 和 gatool() 遗传算法优化的程序界面	遗传算法	360
cwt()	连续小波变换及基小波绘制函数	小波变换	370
dwt()	离散小波变换函数	小波变换	372
idwt()	离散小波反变换函数	小波变换	372
wavemngr()	基小波函数可以由此函数列出	小波变换	373
wavefun()	基小波函数绘制函数	小波变换	373
wavedec()	小波分解函数，可以将信号分解为近似信号与细节信号	小波变换	375

续表

函数名	函 数 功 能	工具箱	本书页码
appcoef()	由分解结果提取近似系数，detcoef() 函数可以提取细节系数	小波变换	375
wrcoef()	由近似系数和细节系数重建信号	小波变换	375
wavemenu()	小波变换工具箱的用户界面主程序	小波变换	377
rslower()	求取粗糙集的下近似集，rsupper() 函数求解上近似集	粗糙集	379
redu()	粗糙集约简函数，core() 函数可以求核集	粗糙集	381
rsdav3()	基于粗糙集理论的数据分析界面主程序	粗糙集	383
fdiffint()	基于 Fourier 级数的分数阶微积分计算函数	自编	386
glfdiff()	基于 Grünwald-Letnikov 定义求取函数分数阶微分的函数	自编	389
ousta_fod()	基于 Oustaloup 算法的分数阶微积分连续滤波器设计函数	自编	392
dfod2()	Ivo Petráš 博士开发的分数阶微积分器的 FIR 滤波器算法函数	—	393
irr_frac()	基于连分数方法的分数阶微积分器的 IIR 滤波器算法函数	自编	394
irr_pade()	基于 Padé 近似方法的分数阶微积分器的 IIR 滤波器算法函数	自编	395
gflode()	基于 Grünwald-Letnikov 定义的分数阶线性微分方程求解函数	自编	397
c10mfode.mdl	分数阶非线性微分方程的搭建模块	自编	399

- 本章介绍了经典集合论问题的 MATLAB 语言求解方法，然后引入了模糊集合的概念，并介绍了基于 MATLAB 语言的模糊集合与模糊推理的实现方法。
- 引入了人工神经网络的数学表示及反馈时神经网络结构，介绍了由 MATLAB 语言进行神经网络结构设置、训练及网络泛化的全过程，并利用 MATLAB 神经网络工具箱直接求解数拟合问题的方法。
- 介绍了遗传算法的基本概念及 MATLAB 求解，并介绍了遗传算法最优化工具箱中的关键函数 gaopt() 及其使用方法，通过例子演示了该函数的各种调用方法和参数设置等。注意，和传统最优化工具箱函数不同，遗传算法最优化工具箱中应该将目标函数定义成最大值。本章还介绍了随着 MATLAB 7.0 版本推出的遗传算法与直接搜索工具箱及其应用。
- 小波分析与变换是目前信号处理及图像处理领域很常用的方法，该方法利用比传统 Fourier 变换更具优势的小波信号对原始信号进行近似与分解。本章介绍了连续、离散小波变换的概念及 MATLAB 求解，介绍了各种基小波波形，还介绍了小波分解与小波重建问题，演示了小波分析技术在信号处理与降噪中的应用。
- 粗糙集理论是近 20 年内才首次提出的理论。本章首先简要介绍了粗糙集理论及要点，引入信息系统决策表的概念与建立，侧重介绍了粗糙集理论问题的 MATLAB 求解，并举例介绍了该理论在条件约简中的应用，还介绍了基于粗糙集理论的图形用户界面程序。
- 本章引入了分数阶微积分的各种定义及求解方法，给出了这些方法的 MATLAB 实现，并给出了未知信号的滤波器设计算法，还分别提出了分数阶线性及非线性微分方

程的数值求解方法，为使用分数阶微积分理论解决实际问题打下了一定的基础。

10.8 习 题

1 考虑一个餐馆小费付费问题[24]。假设平均小费为 15% 消费，试根据服务水平 (例如，可以分为好、中、差或更详细的分段) 和食物质量 (也可以根据实际情况分成若干段) 建立起小费确定的模糊推理系统。

2 已知如下表的样本点 (x_i, y_i) 数据，试利用神经网络理论在 $x \in (1,10)$ 求解绘制出样本对应的函数曲线。还可以尝试不同的神经网络结构和训练算法，将基于神经网络的曲线拟合结果和前面介绍的分段三次多项式插值的算法进行比较。

x_i	1	2	3	4	5	6	7	8	9	10
y_i	244.0	221.0	208.0	208.0	211.5	216.0	219.0	221.0	221.5	220.0

3 假设已知实测数据由下表给出，试利用神经网络对 (x, y) 在 $(0.1, 0.1) \sim (1.1, 1.1)$ 区域内的点进行插值，并用三维曲面的方式绘制出基于神经网络的插值结果。

y_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1
0.1	0.83041	0.82727	0.82406	0.82098	0.81824	0.8161	0.81481	0.81463	0.81579	0.81853	0.82304
0.2	0.83172	0.83249	0.83584	0.84201	0.85125	0.86376	0.87975	0.89935	0.92263	0.94959	0.9801
0.3	0.83587	0.84345	0.85631	0.87466	0.89867	0.9284	0.96377	1.0045	1.0502	1.1	1.1529
0.4	0.84286	0.86013	0.88537	0.91865	0.95985	1.0086	1.0642	1.1253	1.1904	1.257	1.3222
0.5	0.85268	0.88251	0.92286	0.97346	1.0336	1.1019	1.1764	1.254	1.3308	1.4017	1.4605
0.6	0.86532	0.91049	0.96847	1.0383	1.118	1.2046	1.2937	1.3793	1.4539	1.5086	1.5335
0.7	0.88078	0.94396	1.0217	1.1118	1.2102	1.311	1.4063	1.4859	1.5377	1.5484	1.5052
0.8	0.89904	0.98276	1.082	1.1922	1.3061	1.4138	1.5021	1.5555	1.5573	1.4915	1.346
0.9	0.92006	1.0266	1.1482	1.2768	1.4005	1.5034	1.5661	1.5678	1.4889	1.3156	1.0454
1	0.94381	1.0752	1.2191	1.3624	1.4866	1.5684	1.5821	1.5032	1.315	1.0155	0.62477
1.1	0.97023	1.1279	1.2929	1.4448	1.5564	1.5964	1.5341	1.3473	1.0321	0.61268	0.14763

4 De Jong 最优化问题[5]是一个富有挑战性的最优化基准测试问题，其目标函数为

$$J = \min_{\boldsymbol{x}} \boldsymbol{x}^{\mathrm{T}} \boldsymbol{x} = \min_{\boldsymbol{x}} (x_1^2 + x_2^2 + \cdots + x_{20}^2)$$

若 $-512 \leqslant x_i \leqslant 512, i = 1, 2, \cdots, 20$，试用遗传算法得出其最优化问题的解，并用普通的无约束最优化算法函数 `fminunc()` 求解同样的问题，比较两种方法所需的时间和精度。显然，该问题的全局最优解为 $x_1 = x_2 = \cdots = x_{20} = 0$。

5 假设由下面的语句可以生成噪声污染的信号：`>> t=0:0.005:5; y=15*exp(-t).*sin(2*t); r=0.3*randn(size(y)); y1=y+r;` 试利用小波分解与小波重建方法对该信号进行滤波处理，并和第 8 章习题中采用的方法进行比较。

6 试利用遗传算法求解下面的有约束最优化问题，并和传统数值方法进行比较。

$$\max_{\boldsymbol{x}\ \text{s.t.}\begin{cases}0.003079x_1^3x_2^3x_5-\cos^3 x_6\geqslant 0\\ 0.1017x_3^3x_4^3-x_5^2\cos^3 x_6\geqslant 0\\ 0.09939(1+x_5)x_1^3x_2^2-\cos^2 x_6\geqslant 0\\ 0.1076(31.5+x_5)x_3^3x_4^2-x_5^2\cos^2 x_6\geqslant 0\\ x_3x_4(x_5+31.5)-x_5[2(x_1+5)\cos x_6+x_1x_2x_5]\geqslant 0\\ 0.2\leqslant x_1\leqslant 0.5, 14<\leqslant x_2\leqslant 22, 0.35\leqslant x_3\leqslant 0.6,\\ 16\leqslant x_4\leqslant 22, 5.8\leqslant x_5\leqslant 6.5, 0.14\leqslant x_6\leqslant 0.2618\end{cases}} \frac{1}{2\cos x_6}\left[x_1x_2(1+x_5)+x_3x_4\left(1+\frac{31.5}{x_5}\right)\right]$$

7 假设通过实验测出一系列数据，可以列出一个 60×13 的表格，由文件 c10rsdat.txt 给出，其中每行为一个样本，前 12 列每列对应一个条件，最后一列表示某事件是否发生的标志，试用粗糙集的方法找出前 12 个条件中哪些条件对事件的发生起着重要的作用。

8 给定信号 $f(t)=\mathrm{e}^{-3t}\sin(t+\pi/3)+t^2+3t+2$，试利用定义计算出该函数的 0.2 阶微分信号及 0.7 阶积分信号，并将结果信号用曲线表示出来。

9 分别为习题 8 给出的信号设计出连续和离散滤波器，并对该信号进行分数阶微积分运算，和前面介绍出的较精确的数值结果相比较，研究所用方法的精度。

10 考虑第 10.6.2.4 节中介绍的 IIR 滤波器，它可以很好地拟合 Bode 图中的相位特性，但在高频下放大倍数迅速增高，所以对高频噪声有放大作用，这显然不是实际应用中所希望的，试研究在其后设计低通滤波器的方法改善这样的滤波器效果。

11 假设已知分数阶线性微分方程为[38]

$$0.8\mathscr{D}_t^{2.2}y(t)+0.5\mathscr{D}_t^{0.9}y(t)+y(t)=1,\ y(0)=y'(0)=y''(0)=0$$

试求该微分方程的数值解。若将微分阶次 2.2 近似成 2，0.9 阶近似成 1 阶，则可以将该微分方程近似为整数阶微分方程，试比较整数阶近似的计算精度。

12 设分数阶非线性微分方程由图 10-59 中的 Simulink 模型描述，试写出该微分方程的数学表达式，并绘制出输出信号 $y(t)$。

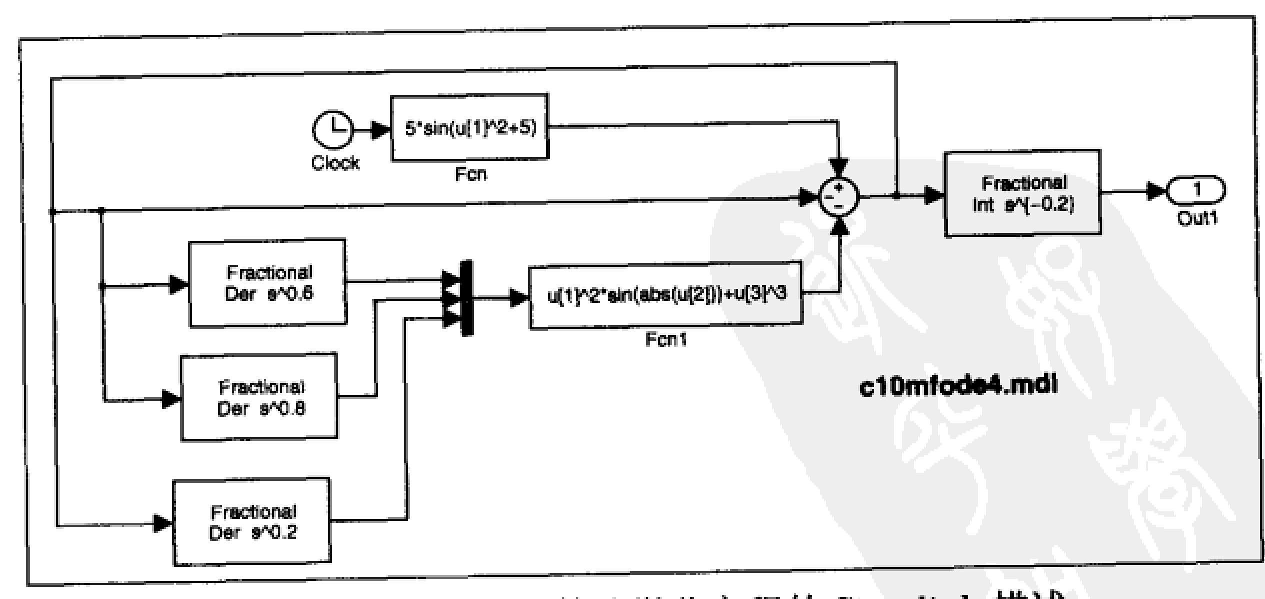


图 10-59 非线性分数阶微分方程的 Simulink 描述

附录A　自由数学语言Scilab简介

本书以MATLAB语言为主要工具介绍了各类数学问题求解方法。从介绍的内容可以看出，用MATLAB现有的工具就可以解决书中提及的所有的问题，如果不能直接求解，还可以借助MATLAB语言本身设计出所需的程序，实现相应的算法。其实，除了MATLAB语言之外，在科学运算领域还有完全免费且公开源代码的Scilab等语言。本附录将简要介绍Scilab语言，使得愿意使用自由软件的读者可以有机会学习这样的语言。Scilab有全套手册的电子版，另外中文版文献[17]是一部很好的起步教材。

第A.1节中将对Scilab的起源、功能和发展作一个简单的概述。第A.2节将介绍Scilab程序设计的基本内容，采用和MATLAB对比的形式介绍，对和MATLAB完全相同的将简单介绍，着重介绍两者的不同之处，先介绍数据结构、语句流程结构等基础知识，再介绍程序开发的基本情况，第A.3节介绍Scilab的二维、三维绘图功能。第A.4节简要介绍可视化仿真环境Scicos。第A.5节介绍Scilab在科学运算方面的应用。本附录中介绍Scilab时除涉及到需要特殊说明的内容外，由于篇幅所限，一般不列出所得的结果，也不给出具体的例子。

A.1　Scilab简介

Scilab语言是法国国家计算机科学与控制研究院INRIA开发的类似于MATLAB的软件，该语言是1989年正式推出的，其源代码完全公开，且为免费传播的自由软件。该语言的主要应用背景是控制与信号处理。Scilab下的Scicos是类似于Simulink的图形化仿真工具。从总体上看，除了其本身独有的个别工具箱外，在语言档次和工具箱的深度与广度上与MATLAB尚有极大差距，但其源代码公开与产品免费这两大特点足以使其成为科学运算研究领域的一种有影响的计算机语言，这里介绍该语言主要为不能获得MATLAB语言或向往自由软件开发的读者入门Scilab之用。

Scilab当前的最新版本是2.7.2，其官方下载网站为

`http://www-rocq.inria.fr/scilab`，　`http://liama.ia.ac.cn/Scilab`

A.2　Scilab的程序设计基础

A.2.1　Scilab变量、常量与数据结构

Scilab下常数的定义与MATLAB有点差异，Scilab中的常数是以百分号`%`引导的，如π表示为`%pi`，$\sqrt{-1}$表示为`%i`；在复数显示上和MATLAB也不同，如`2.4*%i`的显示`2.4i`，而`2*%i`显示为`2.i`，Scilab的注释语句不再用`%`引导，而用`//`直接引导。

Scilab的变量表示方法和MATLAB是一致的，它也支持以双精度浮点数为主的各种

数据结构，如 int32()，int8() 等。但在整型化时，二者的定义是不同的，例如 150 超过了 int8() 表示的范围，所以在 MATLAB 下对其进行截断，故 int8(150) 为 127，而 int8(150) 在 Scilab 转换后的结果为 -106，亦即 $150-2\times128$。

Scilab 支持字符串矩阵，且其表示格式比 MATLAB 简洁，不同长度的字符串可以自动组成字符串矩阵，如表示形式 ['A','BC','EDF';'aaa','abcd','s'] 是可以接受的，而该字符串在 MATLAB 中会给出错误信息，需要事先转换成等长度的字符串。

Scilab 支持多维数组，其定义格式与显示格式与 MATLAB 完全一致。Scilab 同样允许使用 size() 函数和 length() 函数等提取矩阵的维数，也同样允许使用 ***A***(:) 命令提取列向量。

Scilab 支持的列表格式 (list) 类似于 MATLAB 语言中的单元格式，其定义的格式为 ***A***=list(5,[1 2 ; 3 4],'abc')，但在 list() 函数定义下不能定义"矩阵"形式的列表，只能定义一行单元。提取 ***A*** 列表中的第 2 个元素可以用 ***A***(2) 直接完成，而无需像 MATLAB 那样使用大括号。列表格式是可以嵌套的，如 ***A*** 的第 3 列表项可以定义为另一个下级的列表，***A***(3)=list([1 2; 3 4; 5 6],'sbc')，这时可以用 ***A***(3)(1)(3,2) 提取 ***A*** 第 3 列表元素下第 1 列表元素的第 (3,2) 项，即其中的 6。

Scilab 定义了一种常用的多项式数据结构，该结构显然对控制系统分析与设计有益。多项式可以按照其特征根与系数向量分别定义，其中，*p*=poly(***A***,'s') 定义了特征根由 ***A*** 给出的多项式，而 *q*=poly(***B***,'s','c') 定义了多项式系数由 ***B*** 向量给出的多项式。

```
--> p=poly([1 2 3],'s')  // 1,2,3 为多项式的根
p  =
                    2   3
    - 6 + 11s - 6s + s
--> q=poly([1 2 3 4 5 6],'s','c')   // 直接给出系数，注意应该是升幂顺序
q  =
                   2    3    4    5
      1 + 2s + 3s + 4s + 5s + 6s
```

其中 --> 为 Scilab 的提示符，和 MATLAB 语言中的 >> 是一致的。多项式可以直接相乘和相加，得出新的多项式，而无需在相加前像 MATLAB 那样需要将两个多项式先化成等阶次的多项式。例如，用户直接给出 Q=p*q 或 P=p+q 命令，即可以得出多项式的积与和。考虑篇幅，这里不再列出结果了。

表达式 *s*=poly(0,'s') 显然能定义一个根为 $s=0$ 的一阶多项式，亦即算子 s 本身，所以，这样就可以通过计算表示出整个多项式。例如上面的多项式 q 还可以定义为

```
>> s=poly(0,'s'); q=1+2*s+ 3*s^2+ 4*s^3+ 5*s^4+ 6*s^5;
```

A.2.2 Scilab 的基本语句结构

和 MATLAB 语言一样，Scilab 的矩阵赋值也是和 MATLAB 一样的，如

```
-->A=[1 2 3; 2+4*%i 5 6; 7 8 0+6*%i]
A  =
!   1.          2.    3.  !
!   2. + 4.i    5.    6.  !
!   7.          8.    6.i !
```

可以看出，在矩阵的显示方式上稍有不同。Scilab 语言同样支持矩阵的基本运算，也支持像 inv(***A***)这样的函数调用。例如，

```
-->A*inv(A)  // 注意显示格式的不同
 ans  =
!   1. + 1.110E-16i    1.388E-17 - 2.776E-17i  - 5.551E-17 - 1.388E-17i !
!   2.220E-16          1. - 1.110E-16i         - 5.551E-17 + 8.327E-17i !
!   2.220E-16i         1.110E-16 + 2.776E-17i    1. - 1.110E-16i         !
```

A.2.3 Scilab 语言的流程控制语句结构

① 循环结构

和 MATLAB 语言类似，Scilab 的循环结构可以由 for 和 while 引导。这些语句的调用格式分别为

for i=v 循环体 end	while（循环条件） 循环体 end	while（条件）do 或 then 循环体 else 语句组 end

② 转移结构与开关结构

if（条件 1）then 语句组 1 elseif（条件 2）then 语句组 2 ⋮ else 语句组 $n+1$ end	select 变量名 case 表达式 1 then 语句组 1 case 表达式 2 then 语句组 2 ⋮ else 语句组 $n+1$ end

在上面的 while, if, case 等语句中使用的关键词 then 可以由回车键或逗号取代，这样语句的语法结构和 MATLAB 就很接近了。

A.2.4　Scilab 编程

Scilab 函数的语句由 `function` 引导，具体格式与应用均与 MATLAB 完全一致，结束语句为 `endfunction`。函数的内部语句完全可以由 Scilab 语法结构编写，源文件的后缀名为 sci，这样的源程序并不能直接在 Scilab 下执行，而需要用 `exec()` 函数调入。

调入源程序的方法有两种，分别为动态方式和永久方式。动态方式可以使用命令窗口中的 File → Exec 菜单项将编写的 sci 函数调入工作空间，在本次 Scilab 运行中这些函数都是有效的，但若重新启动 Scilab 后就失效了，如果需要该函数则需要重新调入。所谓永久的方法就是将 sci 源程序转换成 bin 文件，置于工作路径下，这样就无需每次 Scilab 均需要重新调入了。

Scilab 还支持一种在线声明的函数定义方法，无需建立 *.sci 文件，可以动态地定义一个函数，直接用于常微分方程组或最优化等问题的求解。动态建立函数可以由 `deff()` 函数实现，具体形式为 `deff('函数引导语句',函数内容字符串)`。例如，想动态建立一个例 7-7 中给出的 `lorenzeq()` 函数，则可以给出如下 Scilab 语句

```
deff('dx=lorenzeq(t,x)',['dx=[-8*x(1)/3+x(2)*x(3);',... // 一行写不下可以续行
    '-10*x(2)+10*x(3); -x(1)*x(2)+28*x(2)-x(3)];'])
```

由 `deff()` 动态定义的函数可以由 `clear` 命令清除，`clear` 命令还可以清除工作空间中的变量。

A.2.5　Scilab 与 MATLAB 的接口

① 某些 MATLAB 语言的函数或程序可以由 Scilab 提供的命令 `mfile2sci` 翻译成 Scilab 程序。然而，由于这两种语言存在很大差异，该命令并不能保证将所有 MATLAB 文件均成功翻译。该命令的调用格式为

`mfile2sci(MATLAB 路径名)`

② MATLAB 下生成的数据可以通过 `mtlb_load` 命令读入 Scilab 环境。该命令的调用格式是很直观的，为

`mtlb_load 数据文件名`

该命令还可以读入 ASCII 格式的数据文件 `mtlb_load 数据文件名 -ascii`。

③ Scilab 还可以将其工作空间中变量存储成 MATLAB 语言可读的文件，这可以通过命令 `mtlb_save` 来实现。该命令的调用格式为

`mtlb_save 文件名 变量列表`

其中，“变量列表”是用空格分隔的需要存储变量的变量名，如 `a b c`。

A.3　Scilab 绘图语句及功能

使用 Scilab 绘图语句时，如果没有打开的图形窗口，则会自动打开一个图形窗口来显示图形。Scilab 的图形窗口有 4 个菜单项，其中 File 是真正的菜单项，允许对图形的读写等，其他 3 个，如 Zoom, UnZoom 与 Rot3D 实质上是按钮，可以对图形开启、关闭局

部放大功能或对三维图进行旋转处理。

和 MATLAB 类似，Scilab 也支持 subplot() 函数对图形窗口进行分割，这样就可以在同一图形窗口中同时绘制出不同的图形了。

A.3.1 二维图形绘制

Scilab 的绘图命令没有 MATLAB 那么丰富，常用的 plot2d() 绘图函数与 MATLAB 语言中的 plot() 函数较接近。其具体的调用格式为

plot2d(t,y,选项)

其中只允许一对 t, y 矩阵，且要求 x 和 y 的维数完全相同。对相同的横坐标 x，MATLAB 用 plot(x,y) 即可以绘制多条曲线，但该调用在 Scilab 下行不同，必须将 x 扩展成多列的矩阵才可以绘制多条曲线，当前版本的 Scilab 也可以使用 plot() 函数，但其调用格式太苛刻了。选项的内容可以由 help plot2d 列出，其中对数坐标曲线绘制可以采用的选项为 logflag，且 logflag='ln' 表示横坐标为对数，纵坐标为线性，而 logflag='nl'。反之，可以通过 n 和 l 的组合来表示坐标轴性质。

Scilab 中默认的绘图方式是在现有的坐标系上叠印新的曲线，相当于直接进入 MATLAB 下的 hold on 状态。若想清除某图形窗口中的所有图形，则可以使用 xbasc(窗口号)，清除当前窗口图形用 sbasc() 即可。还可以通过 xset() 函数修改窗口及坐标系的属性，该函数有点类似于 MATLAB 下的 set() 函数。

二维图形绘制函数 plot2d2() 类似于 MATLAB 中的 stairs() 函数。plot2d3() 绘制的是竖线图，类似于 MATLAB 中的 stem() 函数。

A.3.2 三维图形绘制

三维曲线图形可以由 param3d() 函数绘制，该函数的格式类似于 MATLAB 语言的 plot3() 函数，但其格式要求远严于 plot3() 函数。

三维曲面图形可以由 plot3d(), plot3d1(), plot3d2() 等函数直接绘制，网格图由 plot3d3() 函数绘制，其方法基本上均为 plot3d(x,y,z)，这几个绘制语句绘制出的曲面类型有差异。

【例 A-1】Scilab 生成三维数据的方法与 MATLAB 不同，假设想生成例 2-30 所示的三维曲面图形，则需要给出如下命令：

```
--> x=-3:.1:3; y=-2:.1:2;  // 定义 x,y 网格
    deff('[z]=f(x,y)',...
       ['z1=0.5457*exp(-0.75*y.^2-3.75*x.^2-1.5*x).*(x+y>1);',...
        'z2=0.7575*exp(-y.^2-6*x.^2).*((x+y>-1) & (x+y<=1));',...
        'z3=0.5457*exp(-0.75*y.^2-3.75*x.^2+1.5*x).*(x+y<=-1);',...
        'z=z1+z2+z3;']);  // 定义函数
    z=eval3d(f,x,y); plot3d(x,y,z);  // 函数求值和三维图绘制
```

A.4 Scilab 下的基于模型的仿真方法

Scilab 下还开发了面向框图的系统建模与仿真程序，取名为 Scicos (Scilab Connected Object Simulator)。启动 Scicos 很简单，只需在 Scilab 命令窗口中键入 `scicos()` 即可以打开一个模型编辑窗口，如图 A-1 所示。

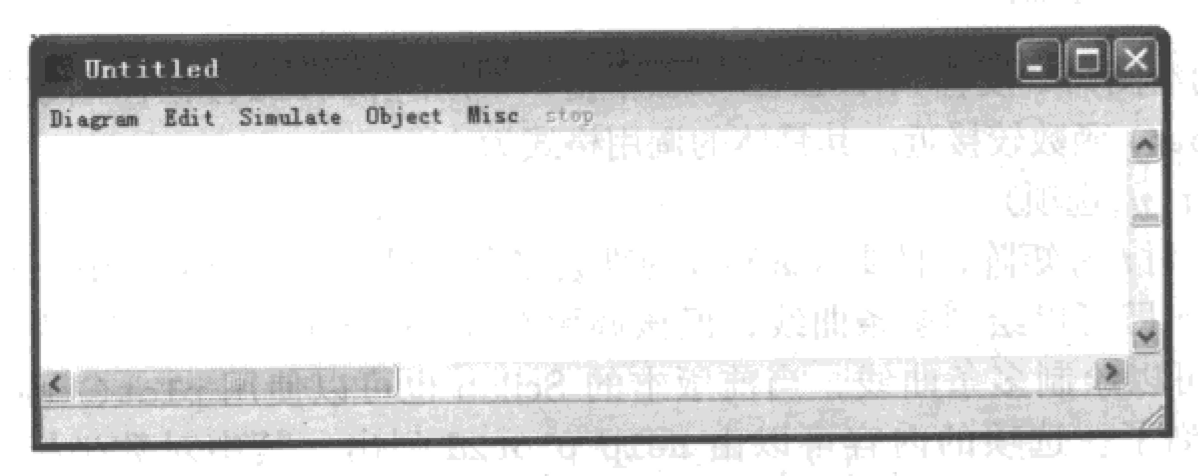
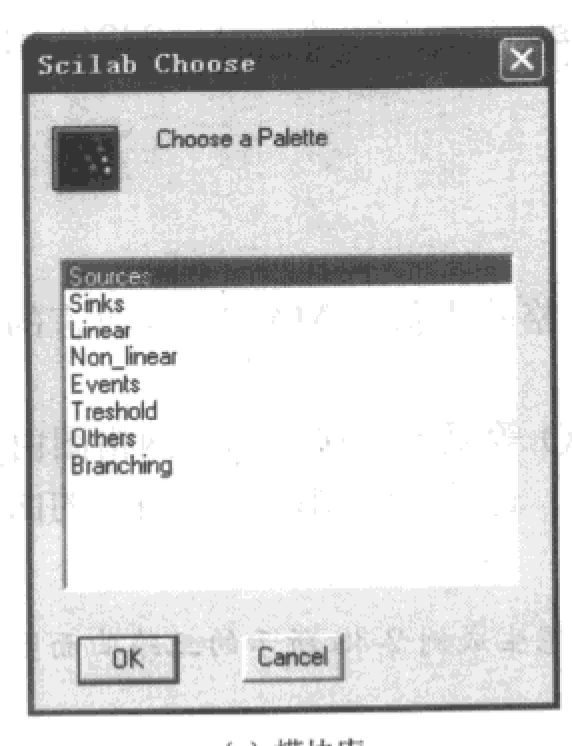
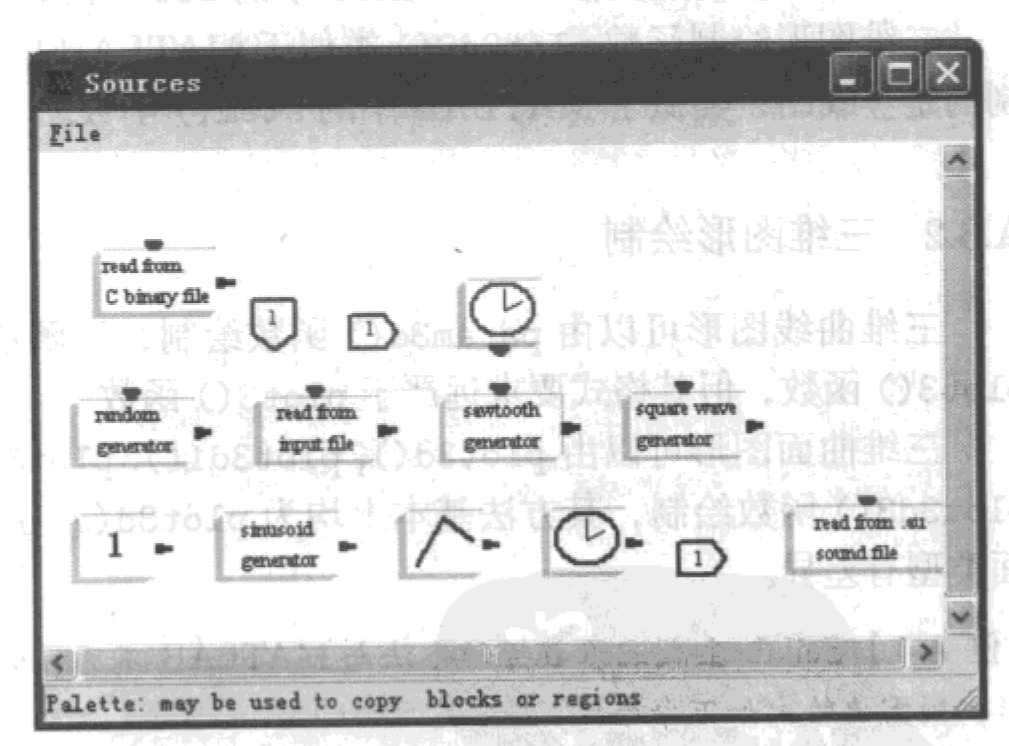


图 A-1 Scicos 空白模型编辑窗口界面

用户还可以选择该界面下的 Edit → Pallette 菜单打开一个对象库窗口，如图 A-2 (a) 所示。和 Simulink 一样，在该模块库中有各个常用的模块组，如其输入源模块组的内容如图 A-2 (b) 所示。所以有 Scicos 程序也可以用图形的方式建立方框图系统的仿真模型，然后调用其中的 Simulation 菜单对之进行仿真分析。



(a) 模块库



(b) 输入源模块组

图 A-2 Scicos 现有模块库与模块组

A.5 基于 Scilab 的数学问题求解

本书介绍 MATLAB 求解数学问题时，很大篇幅是关于解析解或符号运算的，这需要借助于 MATLAB 带的 Maple 引擎，可以利用这样的工具直接调用 Maple 的函数，获得

所需问题的解析解。相比之下，Scilab 这方面的功能要差很多，基本上不能实现灵活的相互调用，只能通过 sci2map() 函数和 maple2sic() 函数，相互转换成 Fortran 源程序，再做成可执行文件来实现相互调用，调用方法很繁杂，且只适合于数值求解，不能进行公式推导。本节也将只介绍基于 Scilab 语言的数学问题数值解方法。

A.5.1 数值微积分问题求解

① 数值微分函数

Scilab 提供了数值差分程序 diff()，可以由 $\boldsymbol{y}$=diff($\boldsymbol{x}$,n) 求出给定向量 $\boldsymbol{x}$ 的 n 阶数值差分。

若已知函数 $f(x)$，则可以用 $\boldsymbol{y}$=numdiff(fun,$\boldsymbol{x}$) 函数求出其数值微分，还可以采用函数 $\boldsymbol{J}$=derivative(fun,$\boldsymbol{x}_0$) 求出多变量函数的 Jacobi 矩阵 $\boldsymbol{J}$。

② 数值积分函数

给定函数 $f(x)$，则可以用 v=intg(a,b,fun) 求出 (a,b) 区间的定积分。这里，fun 仍然可以用前面介绍的 deff() 函数定义出来。另一个函数 integrate() 在求解分段函数积分时也很有特色，该函数的调用格式为 v=integrate(fun,x,a,b)，其中 fun 可以用字符串的格式直接描述被积函数。下面的语句可以得出分段函数在 $(0,\pi)$ 求解的定积分。

```
integrate('if x==0 then 1,else sin(x)/x,end','x',0,%pi)
```

若给出一组实测数据，则可以和样条插值相结合，用 v=intsplin($\boldsymbol{x}$,$\boldsymbol{y}$) 可以较精确地求出该组数据代表函数的定积分。该函数语句格式中默认了 (a,b) 区间值，将直接采用 $\boldsymbol{x}$ 向量的最小和最大值。

二维和三维定积分可以由 int2d() 和 int3d() 函数直接求解。

A.5.2 数值线性代数问题求解

① 矩阵参数化分析

和 MATLAB 语言一样，Scilab 也给出了一组可以用于矩阵参数化分析的函数。例如，d=det($\boldsymbol{A}$), t=trace($\boldsymbol{A}$), d=cond($\boldsymbol{A}$), r=rank($\boldsymbol{A}$) 等函数可以分别计算出 $\boldsymbol{A}$ 矩阵的行列式、迹、条件数与秩，其调用格式与 MATLAB 完全一致。d=norm($\boldsymbol{A}$,选项) 函数也可以和 MATLAB 一样求取矩阵的范数，其选项格式也与 MATLAB 相同。

矩阵的特征值求取函数为 $\boldsymbol{v}$=spec($\boldsymbol{A}$)，函数名和 MATLAB 下的 eig() 函数不同，但调用格式完全一致。该函数同样可以求取矩阵的特征值、特征向量矩阵，并可以求解广义特征值问题。

② 矩阵的分解

常用的矩阵分解，如矩阵的 LU 分解、Cholesky 分解、奇异值分解、QR 分解与正交分解等可以和 MATLAB 一样用 lu()、chol()、svd()、qr() 与 orth() 等函数分别求出，调用格式也基本相同。

③ 矩阵求逆与方程求解

非奇异方阵 $\boldsymbol{A}$ 的逆矩阵可以用语句 $\boldsymbol{B}$=inv($\boldsymbol{A}$) 求出，而奇异矩阵或长方形矩阵

的 Moore-Penrose 广义逆矩阵可以由 $\boldsymbol{B}$=pinv($\boldsymbol{A}$) 求出。线性代数方程 $\boldsymbol{AX}=\boldsymbol{B}$ 与 $\boldsymbol{XA}=\boldsymbol{C}$ 可以用矩阵的左除、右除分别求出，即 $\boldsymbol{X}$=$\boldsymbol{A}$\\$\boldsymbol{B}$ 与 $\boldsymbol{X}$=$\boldsymbol{B}$/$\boldsymbol{A}$，其求解方法与 MATLAB 完全一致。

连续与离散的 Lyapunov 方程均可以由 `lyap()`求出，而在 MATLAB 下需要分别调用 `lyap()` 和 `dlyap()` 函数分别求解。具体表示为

连续 Lyapunov 方程 $\boldsymbol{A}^{\rm T}\boldsymbol{X}+\boldsymbol{XA}=\boldsymbol{C}$，求解方法 $\boldsymbol{X}$=lyap($\boldsymbol{A}$,$\boldsymbol{C}$,'c')

离散 Lyapunov 方程 $\boldsymbol{A}^{\rm T}\boldsymbol{XA}-\boldsymbol{X}=\boldsymbol{C}$，求解方法 $\boldsymbol{X}$=lyap($\boldsymbol{A}$,$\boldsymbol{C}$,'d')

Sylvester 方程在 MATLAB 语言中也可以用 `lyap()` 求解，而在 Scilab 下需要用 `sylv()` 函数来求解，连续与离散 Sylvester 方程均可以调用该函数求解。具体表示为

连续 Sylvester 方程 $\boldsymbol{AX}+\boldsymbol{XB}=\boldsymbol{C}$，求解方法 $\boldsymbol{X}$=sylv($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$,'c')

离散 Sylvester 方程 $\boldsymbol{AXB}-\boldsymbol{X}=\boldsymbol{C}$，求解方法 $\boldsymbol{X}$=sylv($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$,'d')

连续与离散 Riccati 方程均可以由 `ricc()` 函数求解，而在 MATLAB 下需要调用 `are()` 和 `dare()` 函数分别求解连续和离散的 Riccati 方程。在 Scilab 下的求解方法为

连续 Riccati 方程 $\boldsymbol{A}^{\rm T}\boldsymbol{X}+\boldsymbol{XA}-\boldsymbol{XBX}+\boldsymbol{C}=\boldsymbol{0}$，求解方法 $\boldsymbol{X}$=ricc($\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{C}$,'c')

离散 Riccati 方程 $\boldsymbol{X}=\boldsymbol{A}^{\rm T}\boldsymbol{XA}-\boldsymbol{A}^{\rm T}\boldsymbol{XB}(\boldsymbol{D}+\boldsymbol{B}^{\rm T}\boldsymbol{XB})^{-1}\boldsymbol{B}^{\rm T}\boldsymbol{XA}+\boldsymbol{C}$，在求解该方程时应该先给出 $\boldsymbol{F}$=$\boldsymbol{G}$'*inv($\boldsymbol{D}$)*$\boldsymbol{G}$ 命令，再给出求解命令 $\boldsymbol{X}$=ricc($\boldsymbol{A}$,$\boldsymbol{F}$,$\boldsymbol{C}$,'d')。

A.5.3 积分变换与复变函数

由于 Scilab 不支持符号运算，所以很多积分变换的解析解无法解析求出，有理函数的积分变换问题只能通过部分分式展开求出。Scilab 提供了有理函数部分分式展开的函数 `pfss()`，该函数调用格式很直观，为 G_1=pfss(G)，但该函数处理重根时不理想。

【例 A-2】 试求 $G=(s+5)/(s^2+4s+3)$ 和 $G=(s+5)/(s^2+4s+4)$ 的部分分式展开。

【求解】 用下面的依语句可以由 Scilab 立即求出两个函数的部分分式展开为

```
--> s=poly(0,'s');
    G1=(s+5)/(s^2+4*s+3); G=pfss(G1)
 G  =      G(1)          G(2)
            2             - 1
          -----         -----
          1 + s         3 + s
 G  =         G(1)
          5 + s
       ------------
                   2
        4 + 4s + s
```

显然，前一个结果是正确的，而后一个结果有错误。所以在用该部分分式展开的函数时可信度难于保证，应该慎重采用该函数。

A.5.4 最优化问题的求解

Scilab 也提供了各种最优化问题，类似于 MATLAB 的最优化工具箱。这里的最优化问题实际上也是最小化问题，其他最优化问题可以通过相应的变换转换成最小化问题直接求解。Scilab 中直接支持无约束最优化问题求解、线性规划问题求解、二次型规划及一般非线性规划问题求解以及曲线的最小二乘拟合等数学问题的求解。下面分别介绍各类问题的求解方法。

① 非线性最优化问题求解

Scilab 中提供了一个 `optim()` 函数，可以直接求解无约束及有约束一般非线性规划问题，即 $[f,\boldsymbol{x}_{\rm opt}]$=`optim(obj_f,const_f,`$\boldsymbol{x}_0$`)`，其中，目标函数 `obj_f()` 和约束函数 `const_f()` 可以由 `deff()` 动态建立或编写 Scilab 函数来实现，无约束最优化问题可以忽略 `const_f` 参量。对于既有等式约束和不等式约束的问题来说，像 MATLAB 一样，`const_f()` 函数可以返回两个变量，分别表示等式和不等式约束。用户还需要给出初始搜索向量 $\boldsymbol{x}_0$，最后可以通过该函数求解出最优解 $\boldsymbol{x}_{\rm opt}$ 和最优解下的目标函数值 f。

② 线性规划和二次型规划求解

线性规划问题可以由 `linpro()` 函数求解，即 $\boldsymbol{x}_{\rm opt}$ =`linpro(`$\boldsymbol{f},\boldsymbol{A},\boldsymbol{B},\boldsymbol{x}_m,\boldsymbol{x}_M,n_e,\boldsymbol{x}_0$`)`。该函数调用与 MATLAB 下的 `linprog()` 函数有些不同。该函数也同样支持线性等式和不等式约束，在 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 中描述约束条件 $\boldsymbol{Ax} \leqslant \boldsymbol{B}$，其中前面 n_e 个条件描述等式，后面的描述不等式约束。

二次型规划可以由 `quapro()` 函数求解，即 $\boldsymbol{x}_{\rm opt}$ =`quapro(`$\boldsymbol{H},\boldsymbol{f},\boldsymbol{A},\boldsymbol{B},\boldsymbol{x}_m,\boldsymbol{x}_M,n_e,\boldsymbol{x}_0$`)`，其说明类似于线性规划函数。

A.5.5 微分方程的数值解

① 一般常微分方程组的数值解

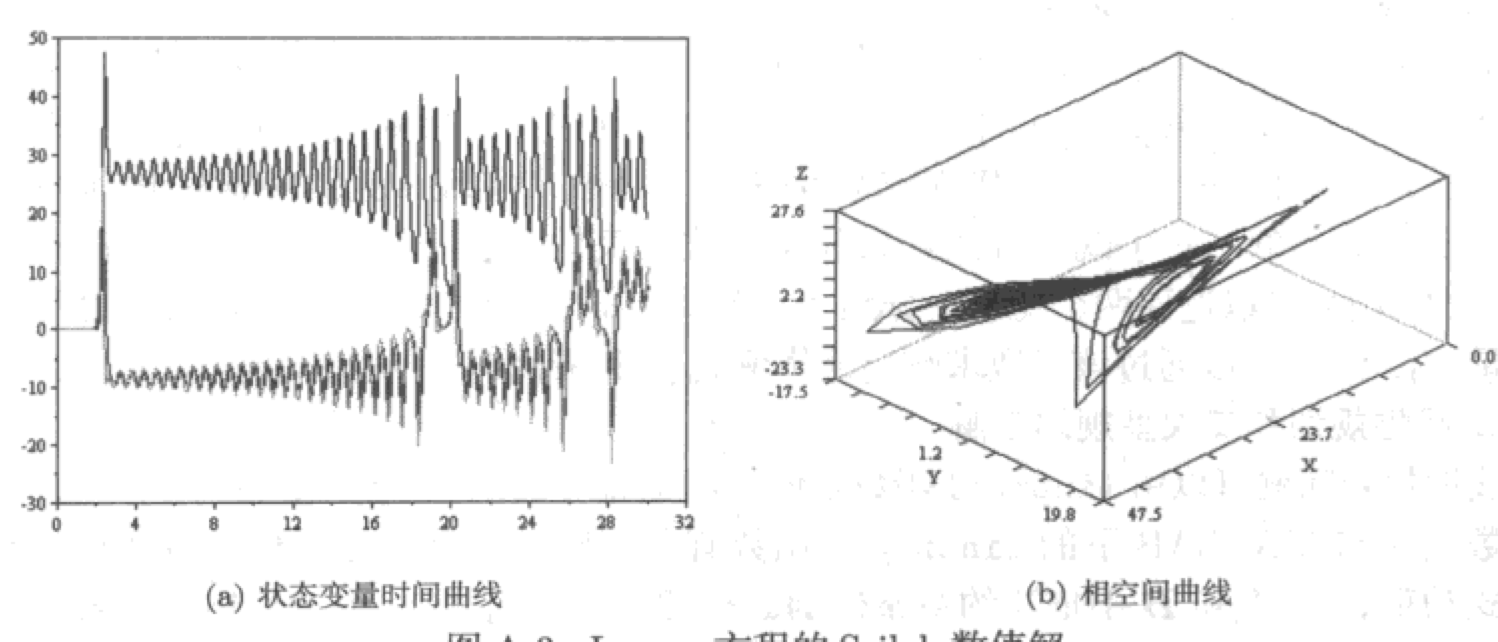
Scilab 中提供了一个 `ode()` 函数，可以用于一般一阶微分方程组的数值求解。其函数下实现了各种微分方程数值解算法，故该函数相当于 MATLAB 下的 `ode45()`, `ode15s()` 等函数。该函数的调用格式为 $\boldsymbol{x}$=`ode(`$\boldsymbol{x}_0$, t_0, $\boldsymbol{t}$, `fun)`，其中，$\boldsymbol{x}_0$ 为初始状态向量，t_0 为初始时刻，$\boldsymbol{t}$ 为用户选定的时间向量，但这不意味着直接计算这些点处的状态变量值，它们直接还可以根据误差限的要求选择更密的点，但返回的只是这些点处的状态向量值 $\boldsymbol{x}$。`fun` 是描述相应一阶微分方程组的 Scilab 函数，它可以用 `deff()` 函数动态建立。微分方程求解的选项可以用 `odeoptions()` 函数来进一步设置，其中可以选定各种算法、误差限等。下面通过例子来演示微分方程求解方法。

【例 A-3】仍考虑例 7-7 中给出的 Lorenz 方程问题，假设初始状态变量为 $\boldsymbol{x}_0$=`[0;0;1e-10]`，且时间向量为 $\boldsymbol{t}$=`[0:.1:30]`，则可以利用动态建立起来的描述该方程的 Scilab 函数 `lorenzeq()` 求解微分方程，并绘制其时间响应曲线及三维相空间曲线。

```
--> deff('dx=lorenzeq(t,x)',['dx=[-8*x(1)/3+x(2)*x(3);',...
       '-10*x(2)+10*x(3); -x(1)*x(2)+28*x(2)-x(3)];']);  // 定义函数
    x0=[0;0;1e-10]; t=0:.05:30;
```

```
x=ode(x0,0,t,lorenzeq);  plot2d([t' t' t'],x')
--> param3d(x(1,:),x(2,:),x(3,:)) // 绘制三维曲线
```

得出的时间响应曲线和相空间曲线分别如图 A-3 (a)、图 A-3 (b) 所示。由得出的三维相空间曲线可见，很多地方有很大跳跃，这是因为选定的 t 向量有的地方相对解本身来说跳跃过大，不能得出平滑语句，所以人为选择 t 的方法并非很合适。可见，ode() 函数的调用格式过于生硬，远没有 MATLAB 的相关语句调用那么实用。



(a) 状态变量时间曲线　　(b) 相空间曲线

图 A-3　Lorenz 方程的 Scilab 数值解

② 微分代数方程组的数值解

Scilab 也提供了微分代数方程 $\boldsymbol{M}(t,\boldsymbol{x})\dot{\boldsymbol{x}}(t)=\boldsymbol{f}(t,\boldsymbol{x})$ 的求解函数，如采用 dassl() 函数求解该方程，其调用格式为 $\boldsymbol{x}$=dassl($\boldsymbol{x}_0$,t_0,$\boldsymbol{t}$,fun,Mfun)。其中，fun 和 Mfun 为微分代数方程的两个函数。函数 dasrt() 和 impl() 也能实现微分代数方程的数值求解。

③ 常微分方程组边值问题的数值解

微分方程边值问题可以调用 Scilab 的 bvode() 函数求解。

A.5.6　数据处理的实现

① 线性插值函数

线性插值可以由 $\boldsymbol{y}$=interpln($\boldsymbol{X}$,$\boldsymbol{x}$) 函数求解，其中，$\boldsymbol{X}=[\boldsymbol{x}_0,\boldsymbol{y}_0]^{\mathrm{T}}$ 为两行矩阵，$\boldsymbol{x}_0$ 为样本点的横坐标，$\boldsymbol{y}_0$ 为纵坐标，$\boldsymbol{x}$ 为插值点的横坐标，$\boldsymbol{y}$ 为线性插值结果。

② 样条插值函数

样条插值可以用 interp() 函数计算，即 $\boldsymbol{y}$=interp($\boldsymbol{x}$,$\boldsymbol{x}_0$,$\boldsymbol{y}_0$,$\boldsymbol{d}$)。其中，$\boldsymbol{x}_0,\boldsymbol{y}_0$ 为已知样本点数据，$\boldsymbol{x}$ 为插值点的横坐标，$\boldsymbol{d}$ 为样条函数 splin() 的计算结果，$\boldsymbol{d}$=splin($\boldsymbol{x}_0$,$\boldsymbol{y}_0$)，这时 $\boldsymbol{y}$ 为插值结果。

③ 曲线平滑计算

给定一组实验数据 $\boldsymbol{x}_0$，$\boldsymbol{y}_0$，则可以通过 $\boldsymbol{Y}$=smooth($\boldsymbol{X}$) 对其自动平滑处理。其中，$\boldsymbol{X}=[\boldsymbol{x}_0,\boldsymbol{y}_0]$，得出的 $\boldsymbol{Y}$ 也是两列矩阵，分别表示平滑后的横纵坐标。

④ 曲线的最小二乘拟合

曲线拟合有点像 MATLAB 最优化工具箱中的 `lsqcurvefit()`，要求用户先建立一个原型函数 `fun()`，则可以由 $[f, x_{\rm opt}]$=`leastsq(fun,`x_0`)` 进行最小二乘曲线拟合。

A.5.7 概率论与数理统计

Scilab 也有自己的统计学工具箱，提供了类似于 MATLAB 统计学工具箱的部分内容，但远没有 MATLAB 工具箱那么强大。这里将介绍该工具箱的一些函数。

① 伪随机数生成

用 $\boldsymbol{v}$=`rand(`n,m`,'uniform')` 函数将生成一个 $n \times m$ 的 $(0,1)$ 均匀分布伪随机数矩阵 $\boldsymbol{V}$；用 $\boldsymbol{v}$=`rand(`n,m`,'normal')` 将生成一个 $n \times m$ 的 $N(0,1)$ 标准正态分布伪随机数矩阵 $\boldsymbol{V}$。

② 均值与方差计算

类似于 MATLAB 语言，一个随机数向量 $\boldsymbol{x}$ 的均值可以由 μ=`mean(`$\boldsymbol{x}$`)` 函数求出，用 s=`var(`$\boldsymbol{x}$`)` 和 s=`stdev(`$\boldsymbol{x}$`)` 函数可以求出其方差和标准方差，用 $\boldsymbol{V}$=`mvvacov(`$\boldsymbol{X}$`)` 函数求取 $\boldsymbol{X}$ 矩阵的协方差矩阵，用 `correl()`函数还求出两个随机变量的相关系数。

用 m=`moment(`$\boldsymbol{x}$,n`)` 函数可以求出 $\boldsymbol{x}$ 向量的 n 阶原点矩，而用 m=`cmoment(`$\boldsymbol{x}$,n`)` 函数可以求出 $\boldsymbol{x}$ 向量的 n 阶中心矩。

A.6 本章要点简介

本附录较全面地介绍了 MATLAB 的一个替代软件系统 —— Scilab，该软件是自由软件，全部源代码是公开的。附录还简要介绍了该语言在程序设计、科学可视化等方面的应用以及该语言在数学问题求解中的应用，包括数值微积分问题、数值线性代数问题、方程求解与最优化问题、微分方程求解问题、数据处理问题及概率论与数理统计中的应用。然而，本书用 MATLAB/Maple 求解的很大一类问题，其数学问题的解析求解问题是无法用现有的 Scilab 语言直接求解的。

从现有的水平看，Scilab 远远没有达到 MATLAB 的水平，在很多领域还应该称为初级阶段。但它的出现与流行，为自由软件的爱好者还是提供了一种解决问题的手段，有一定的应用价值。

A.7 习 题

1 用 Scilab 语句绘制出 $y(x) = \sin(1/t)$ 在 $t \in [-\pi, \pi]$ 区间内的曲线。

2 对下面给出的各个矩阵求取各种参数，如矩阵的行列式、迹、秩、特征多项式、范数、逆矩阵、特征值与特征向量等，并求出 $\mathrm{e}^{\boldsymbol{A}}$、$\mathrm{e}^{\boldsymbol{B}}$。

$$\boldsymbol{A} = \begin{bmatrix} 7.5 & 3.5 & 0 & 0 \\ 8 & 33 & 4.1 & 0 \\ 0 & 9 & 103 & -1.5 \\ 0 & 0 & 3.7 & 19.3 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix}$$

3 求解下面的 Sylvester 方程，并检验所得解的精度。

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} \boldsymbol{X} + \boldsymbol{X} \begin{bmatrix} 2 & 3 & 6 \\ 3 & 5 & 2 \\ 3 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 4 & 1 \\ 5 & 2 & 1 \end{bmatrix}$$

4 用数值方法求解下面的常微分方程。

$$\dot{\boldsymbol{y}} = \begin{bmatrix} -21 & 19 & -20 \\ 19 & -21 & 20 \\ 40 & -40 & -40 \end{bmatrix} \boldsymbol{y},\ \boldsymbol{y}_0 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

5 求解下面的有约束最优化问题。

$$\min_{\boldsymbol{x} \text{ s.t.} \begin{cases} x_1^2+x_2^2+x_3^2-25=0 \\ 8x_1+14x_2+7x_3-56=0 \\ x_1,x_2,x_3 \geqslant 0 \end{cases}} \left[1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3\right]$$



参考文献

[1] Atherton D P, Xue, D. The analysis of feedback systems with piecewise linear nonlinearities when subjected to Gaussian inputs. In: Kozin F, Ono T (eds.). Control systems, topics on theory and application, pp23-38. Tokyo: Mita Press, 1991

[2] Bosley M J, Kropheller H W, and Lees F P. On the relation between the continued fraction expansion and moments matching methods of model reduction. International Journal of Control, 18:461-474, 1973

[3] 陈传璋，金福临，朱学炎等. 数学分析. 北京：人民教育出版社，1979

[4] Chen Y Q, Vinagre B M. A new IIR-type digital fractional order differentiator. Signal Processing, 83:2359-2365, 2003

[5] Chipperfield A, Fleming P, *et al.* Genetic algorithm toolbox user's guide. Department of Automatic Control and Systems Engineering, University of Sheffield, 1994

[6] Conover W J. Practical nonparametric statistics. New York: Wiley, 1980

[7] Dongarra J J, Bunsh J R, Molor C B. LINPACK user's guide. Philadelphia: Society of Industrial and Applied Mathematics (SIAM), 1979

[8] Forsythe G E, Malcolm M A, Moler C B. Computer methods for mathematical computations. Englewood Cliffs: Prentice-Hall, 1977

[9] Frank G. The Maple Book, Chapman & Hall/CRC, 2001

[10] Garbow B S, Boyle J M, Dongarra J J, Moler C B. Matrix eigensystem routines — EISPACK guide extension, Lecture notes in computer sciences. Vol. 51, New York: Springer-Verlag, 1977

[11] Gongzalez R C, Woods R E. Digital image processing (2nd ed). Englewood Cliffs: Prentice-Hall, 2002 (电子工业出版社有影印版)。

[12] Hagan M T, Demuth H B and Beale M H. Neural network design. PWS Publishing Company, 1995，中译本：戴葵等译. 神经网络设计. 机械工业出版社，2002

[13] 何文章，桂占吉，贾敬. 大学数学实验. 哈尔滨：哈尔滨工程大学出版社，1999

[14] Hilfer R. Applications of fractional calculus in physics. Singapore:World Scientific, 2000

[15] Holmström K, Göran A and Edvall M M. User's guide for TomLAB 4.1.0，电子版，2000

[16] Houck C R, Joines J A, Kay M G. A genetic algorithm for function optimization: a MATLAB implementation. GAOT 工具箱手册电子版，1995

[17] 胡包钢，赵星，康孟珍. 科学计算自由软件 SCILAB 教程. 北京：清华大学出版社，2003

[18] 胡运权主编. 运筹学教程. 北京：清华大学出版社，1998

[19] 黄琳. 系统与控制理论中的线性代数. 北京：科学出版社，1984

[20] Knuth D E. The TEXbook: Volumn A of computers and typesetting. Reading MA: Addison-Wesley Publishing Company, 1986

[21] Lamport L. LATEX: a document preparation system — user's guide and reference manual. Reading MA: Addision-Wesley Publishing Company, 2nd edition, 1994

[22] 李岳生，黄友谦. 数值逼近. 北京：人民教育出版社，1978

[23] 陆璇. 应用统计. 北京：清华大学出版社，1999

[24] MathWorks. Fuzzy logic toolbox user's manual, 2004

[25] MathWorks. Genetic algorithm and direct search toolbox user's manual, 2004

[26] MathWorks. Signal processing toolbox user's manual, 2004

[27] MathWorks. Using MATLAB (MATLAB 用户手册), 2004

[28] MathWorks. Simulink user's manual. 2004

[29] Moler C B. MATLAB—An Interactive Matrix Laboratory. Technical Report 369, Department of Mathematics and Statistics, University of New Mexico, 1980.

[30] Molor C B. Numerical Computing with MATLAB. MathWorks Inc, 2004

[31] Moler C B, Stewar G W. An algorithm for generalized matrix eigenvalue problems. SIAM Journal of Numerical Analysis. 10:241-256, 1973

[32] Moler C B, Van Loan C F. Nineteen dubious ways to compute the exponential of a matrix. SIAM Review, 20: 801-836, 1979

[33] Nelder J A, Mead R. A simplex method for function minimization. Computer Journal, 7: 308-313, 1965

[34] Numerical Algorithm Group. NAG FORTRAN library manual. 1982

[35] Oustaloup A, Levron F, Mathieu B, Nanot F M. Frequency-band complex noninteger differentiator: characterization and synthesis. IEEE Transaction on Circuit and Systems-I: Fundamental Theory and Applications, 47(1):25-39, 2000

[36] Pawlak Z. Rough sets — theoretical aspects of reasoning about data. Boston: Kluwer Academic Pub., 1991

[37] Petráš I, Podlubny I, O'Leary P *et al.* Analogue realization of fractional order controllers. Fakulta BERG, TU Košice, 2002

[38] Podlubny I. Fractional differential equations. San Diago: Academic Press, 1999

[39] Press W H, Flannery B P, Teukolsky S A, and Vitterling W T. Numerical recipes, the art of scientific computing. Cambridge: Cambridge University Press, 1986

[40] 邵军力，张景，魏长华. 人工智能基础. 北京：电子工业出版社，2000

[41] 《数学手册》编写组. 数学手册. 北京：人民教育出版社，1979

[42] Smith B T, Boyle J M, Dongarra J J *et al.* Matrix eigensystem routines – EISPACK guide, Lecture notes in computer sciences. Vol. 6 (Second edition). New York: Springer-Verlag, 1976

[43] Tseng C-C, Pei S-C, Hsia S-C. Computation of fractional derivatives using Fourier transform and digital FIR differentiator. Signal Processing, 80:151-159, 2000

[44] 汪培庄. 模糊集合论及其应用. 上海：上海科学技术出版社，1983

[45] 王小平，曹立明. 遗传算法 —— 理论、应用与软件实现. 西安：西安交通大学出版社，1998

[46] 王珏. Rough set 约简与数据浓缩. 高技术通讯，11:40-45, 1997

[47] Vinagre B M and YangQuan Chen. Fractional calculus applications in automatic control and robotics. 41st IEEE CDC, Tutorial workshop 2, Las Vegas, 2002.

[48] Wolfram S. The Mathematica book. Cambridge University Press, 1988

[49] 武汉大学，山东大学. 计算方法. 北京：人民教育出版社，1979

[50] 西安交通大学高等数学教研室编. 复变函数(第二版). 北京：人民教育出版社，1981

[51] Xue D. Model reduction approaches for control systems. 东北大学讲义，1994

[52] 薛定宇，陈阳泉. 基于 MATLAB/Simulink 的系统仿真技术与应用. 北京：清华大学出版社，2002

[53] Zadeh L A. Fuzzy sets. Information and Control, 8:338-353, 1965

[54] 张化光，王智良，黄伟. 混沌系统的控制理论. 沈阳：东北大学出版社，2003

[55] 张雪峰. 粗糙集数据分析系统应用平台的研究与程序开发. 沈阳：东北大学硕士学位论文，2004

[56] 赵选民，师义民. 概率论与数理统计典型题分析解集. 西安：西北工业大学出版社，1999

[57] 中山大学数学力学系. 概率论与数理统计. 北京：人民教育出版社，1980